View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by NORA - Norwegian Open Research Archives

2009

MASTER'S THESIS

Ontology Based Natural Language Discovery of Semantic Web Services in the Geospatial Domain

Yuzhu Yang

Acknowledgement

I would like to take this opportunity to thank my supervisor: Dr. Arne Jørgen Berre from SINTEF and UiO. This thesis would not have been possible without his support, as he has been extremely helpful in leading me into the whole picture of the problem domain and provided me a great deal of flexibility in carrying out my work. He has also been very encouraging and patient in the whole process. Dr. Diana Santos from SINTEF provided a valuable introduction to Question and Answering (QA) Systems in the beginning of my thesis work. I am also grateful to SINTEF researcher Luis Fernando Costa in helping me with understanding the concepts in SWING project which is relevant to my thesis research.
I should also thank my school-mates for working together with me and encouraging me, special thanks to my school-mate Yun Huang for great encouragement and company during the last thesis writing phase. Finally, my family, my friends Hejia and Jiafei deserve special mention: their belief in me is very valuable.

Abstract

Semantic Web Services (SWS) has been an active research topic in the recent years. It aims to achieve automatic Web service discovery, and invocation and composition of services, by adding semantics to the services, which is very significant for accelerating the achievement of Service-oriented architectures by solving a lot of interoperability issues. Geospatial data on the Web contains more and more important information. Capturing, analyzing and managing such data can help people to achieve various kinds of tasks. Discovery of Semantic Web Services in Geospatial domain is therefore significant. Nowadays the existing semantic Web service discovery frameworks in geospatial domain have either complex interface or require extra knowledge (for instance, the Web Service Modeling Language (WSML) to be able to provide correct queries.

This thesis suggests an Ontology Based Natural Language Discovery of Semantic Web Services (ONALDIS). ONALDIS works generally in the geospatial domain, and offers a command line interface which takes natural language questions as input, and gives a WSML query as output. This WSML query can be further processed by an existing SWS discovery framework to discover semantic Web services. ONALDIS achieves the mapping between natural language query and WSML query by analyzing domain ontologies with some rules and generates word sets and WSML queries from the ontologies.

In this thesis, semantic Web services and geospatial domain are first introduced. The SWING system and the need for a natural language interface are then discussed. Some existing approaches for natural language interfaces, mainly Question Answering (QA) systems are presented. Further, ONALDIS is introduced as a solution of a natural language interface for semantic Web service discovery in the geospatial domain, with an implementation for the SWING system. Finally, the conclusions are presented and further work is suggested.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

This chapter provides an introduction to this thesis including background and motivation, goal and approach.

1.1 Background and Motivation

Nowadays geographical/geospatial data on the Web contains more and more important information. Capturing, analyzing and managing these data can help people to achieve various kinds of tasks. These Geographical data is distributing on the Web in an irregular way which makes data capturing and analyzing process not that trivial. GIS applications are widely used to create interactive queries, analyze spatial information, edit data, view maps, and present the results of all these operations. A lot of queries in geospatial domain can be expressed by natural language. One example is about a quarry in France. The information relevant to this quarry is illustrated in Figure 1-1 Quarry in France

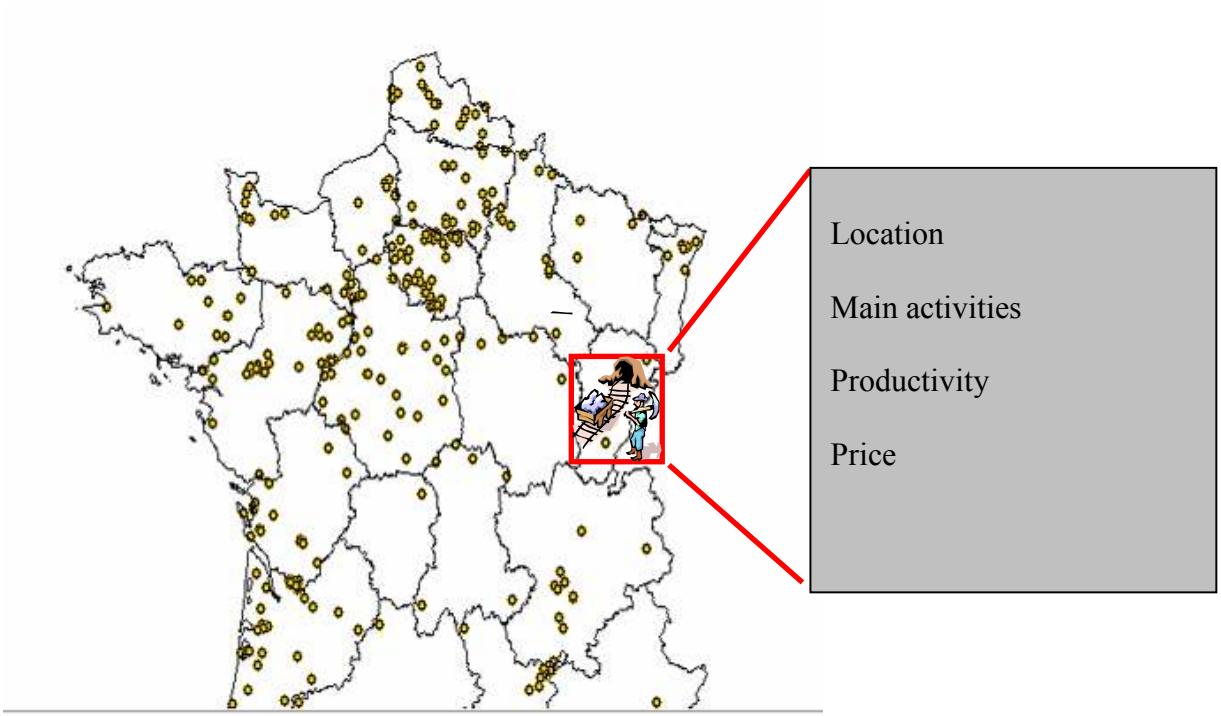


Figure 1-1 Quarry in France

Some questions relevant to this can be listed as follows:
- Where are the quarries in France?
- What quarries in France can produce over 100 000 tons per year?
- What is the lowest price for one ton cement in France?
- What are the main activities of constructing a road?

To find the answers or useful information to these questions, a Web user will normally choose the following approaches:

- Through some websites they already knew from before. Follow the navigation of those websites.

- Through traditional search engine such as Google, Yahoo etc.
- Through Question Answering Systems.
- Through (Semantic) Web services.
- Etc.

The different approaches mentioned above have both advantages and disadvantages. The first three approaches are easier to achieve, but they miss the resource from Web services part. The last approach is about (Semantic) Web services. Nowadays Web services are gaining more and more interest. The popular of SOA (Service Oriented Architecture) further pushes this process forward. Semantic Web services technology is developed based on Web services and is a component of Semantic Web [42]. Thus the technologies for discovery, combining and invoking semantic Web services are so far too complex for Web users and even for domain professionals. Therefore a simpler interactive solution is demanded. With the development of semantic Web services, a few frameworks for semantic Web services discovery, composition and invocation are emerging to meet this demand. For instance, SWING [43] and SAP's Guided Procedures (GP) [44]. The user interface for semantic Web services discovery of these frameworks is not convenient for end users. In this thesis SWING framework is used as a base to illustrate how to provide user a convenient interface to process natural language questions which helps to discovery semantic Web services. This is the main topic to be addressed in this thesis work. Since SWING project has the geospatial as the ontology domain, the thesis work also focuses on geospatial domain. It is possible to be adjusted to apply a broader domain. Figure 1-2 illustrates where the topic of this thesis exists in the whole web picture.

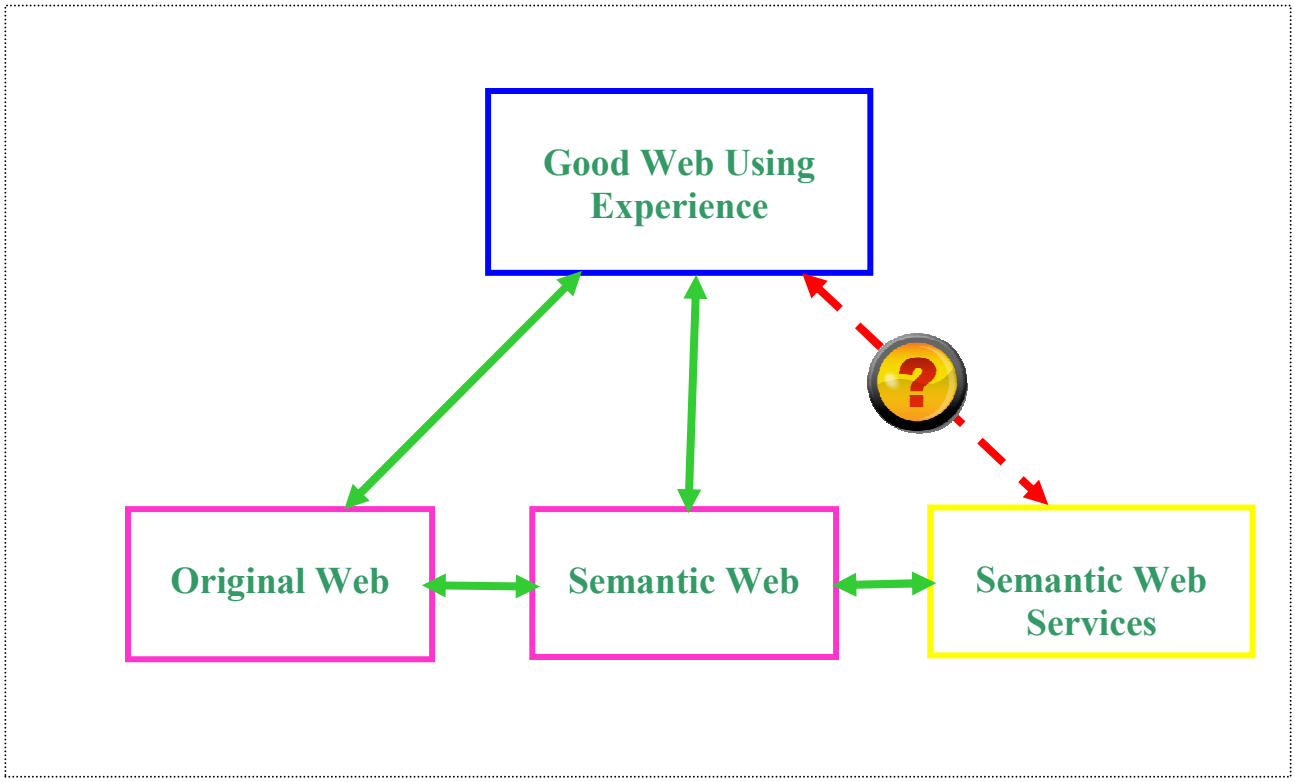


Figure 1-2 Web Using Experience

1.2 Goal and Approach

The goal of the thesis is to provide user a convenient interface based on semantic Web services processing framework, to process natural language questions which helps to discover semantic Web

services in geospatial domain. With the help of ontologies, the interface is supposed to take queries in natural language as input and generate WSML queries which can query SWS in geospatial domain in order to simplify the querying process to SWS. SWING system is used as a semantic Web services processing framework in this thesis.

In order to address this goal, a number of questions are listed to help to understand the relevant concepts and researches that have been done in this problem area.

1. What is the problem with the current discovery of semantic Web services in geospatial domain?
2. What is the problem with the current SWING interface?
3. How to create a natural language interface for SWING system to discover semantic Web services in geospatial domain?
4. Is the natural language interface also suitable for other domains besides SWING?

Chapter 2 Method of Work

The main research methods employed within this thesis are Hypothetic-Deductive method [27] and Technology research. The Hypothetic-Deductive method is used on addressing suggesting a solution for the main problem that this thesis wants to solve. Technology Research is used as the main research process.

In chapter 2.1, the Hypothetic-Deductive method is introduced and how it is applied on the thesis work is discussed. In chapter 2.5, Technology Research is introduced. Chapter 2.3 discusses why the two research methods are chosen to be used comparing to other research methods, and why they are proper research methods for this thesis work.

2.1 Hypothetic-Deductive Method

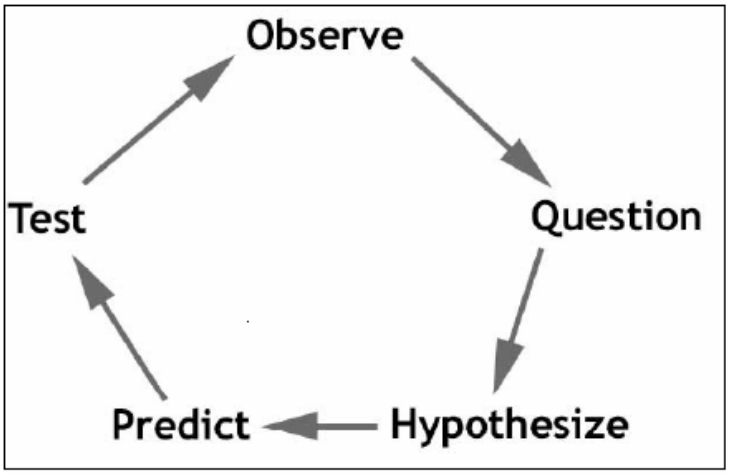


Figure 2-1 Key Elements in Hypothetic-Deductive Method

Hypothetic-Deductive method is the most important research method used in the thesis. Figure 2-1 shows the key elements in Hypothetic-Deductive method. It was first named by William Whewell, and later popularized after Karl Popper's citation of the term. It is a very important method for testing theories or hypotheses. According to it, scientific inquiry proceeds by formulating a hypothesis in a form that could conceivably be falsified by a test on observable data. A test that could and does run contrary to predictions of the hypothesis is taken as a falsification of the hypothesis. A test that could but does not run contrary to the hypothesis corroborates the theory. It is then proposed to compare the explanatory value of competing hypotheses by testing how stringently they are corroborated by their predictions.

The application of Hypothetic-Deductive method can be divided into four stages:

- Identify the hypothesis to be tested.
- Generate predications from the hypothesis.
- Use experiments to check whether predictions are correct.
- If the predictions are correct, then the hypothesis is confirmed. If not, then the hypothesis is dis-confirmed.

The hypothesis specifies the assumptions regarding the solution to the problem, and will serve as guidance to validate the results. The high level hypothesis in the context of thesis is:
It is possible to create an ontology based natural language interface for discovery of SWS in the geospatial domain.

In Table 2-1 the hypotheses are divided into several sub-hypotheses, which will be used as a basis to validate the results of this thesis. The test cases associated with the hypotheses are presented in the later chapter.

	Hypothesis
H	It is possible to create an ontology based natural language interface for discovery of SWS in the geospatial domain.
H 1.1	It is possible to create a system with the help of ontologies which takes NL questions as input and generate WSML queries which can query SWS in geospatial domain in order to simplify the querying process to SWS.
H 1.1.1	It is possible to create a system with the help of ontologies which takes NL questions as input and generate WSML queries which can query SWS in SWING domain.
H 1.1.2	H1.1.1 can be applied to different ontologies domains.
H 1.1.3	The accuracy of the answer through the natural language interface is as good as using a specialized query language.
H 1.1.4	Natural language interface is easier to use for end users.
H 1.2	The output of H1.1 can be the input of some Semantic Web services frameworks which can discover SWS in geospatial domain.

Table 2-1 Hypothesis

2.2 Technology Research

According to Solheim and Stølen [1], Technology Research is an iterative process with the following main steps:

- **Problem analysis** – find a problem to which a solution is needed by interacting with possible users and other stakeholders.
- **Innovation** – construct an artifact that satisfies the potential need.
- **Evaluation** – based on the potential need, formulate predictions about the artifact and checks whether these predictions come true. If the predictions turn out to be correct, it can be argued that the artifact solves the identified problem.

The three steps in the thesis context are further described in the following sections.

2.2.1 Problem Analysis

As mentioned in the introduction chapter, to find the answers or useful information on Web, a Web user will normally choose one of the following approaches:

- Through some websites they already knew from before. Follow the navigation of those websites.
- Through traditional search engine such as Google, Yahoo etc.
- Through Question Answering Systems.
- Through (Semantic) Web services.

From these approaches, it shows that the resources on Web are mainly separated into two categories, data and services, as illustrated in Figure 2-2.

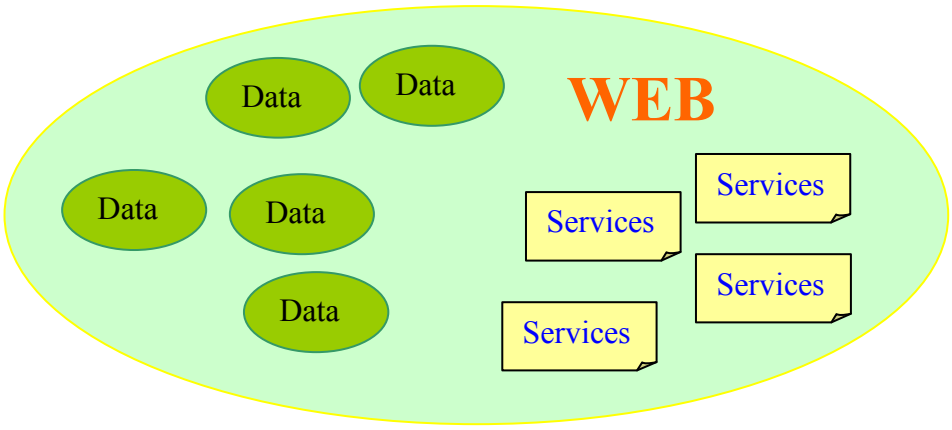


Figure 2-2 Web Resources

Data here means the collection of Web resources, including those web page content, for instance an article, an image, a piece of video, and those data resources, for instance, a database record. A Web Services is any service that is available over the Internet, uses a standardized XML messaging system, and is not tied to any one operating system or programming language. It is usually registered and published in service registries. The Semantic Web services are not used widely. One reason is that the technologies for discovery, combining and invoking semantic Web services are so far too complex for Web users and even for domain professionals. There exist a few frameworks for semantic Web services discovery, composition and invocation come out with demand. For instance, SWING system is one of them. But the user interface of it is still complex. For instance, SWING system takes WSML query as input, where WSML is a new programming language which cannot be easily understood.

Therefore a simpler interactive solution is demanded.

The Problem analysis part is related to chapter 4, Swing System and the Need for Natural Language Interface, and chapter 5, Analysis of Different Approaches for Natural Language Interface.

2.2.2 Innovation

The innovation of this thesis is to create an ontology based natural language interface for discovery of SWS in the geospatial domain. The system called ONALDIS (Ontology Based Natural Language Discovery of Semantic Web Services) illustrates how to provide user a convenient interface to process natural language questions which helps to discovery semantic Web services. The innovation part is related to chapter 6, Ontology Based Natural Language Discovery of Semantic Web Services (ONALDIS).

2.2.3 Evaluation

The evaluation of the thesis is based on the hypotheses presented in Table 2-1. ONALDIS is applied on SWING system to evaluate its performance.
This part is related to chapter 7 ONALDIS Applied on SWING.

2.3 Research Methods and Thesis Structure

The Hypothetic-Deductive method is used on addressing an analysis for solution for the main problem that this thesis wants to solve. Technology Research is used as the main research process.
The relationship between the two research methods and the structure of the thesis is illustrated in Figure 2-3. Chapter 4 and chapter 5 is problem analysis phase, which correspond to Observe, Question and Hypothesize. Chapter 6 is Innovation phase, which correspond to Predict. Chapter 7 is Evaluation phase, which corresponds to Test.

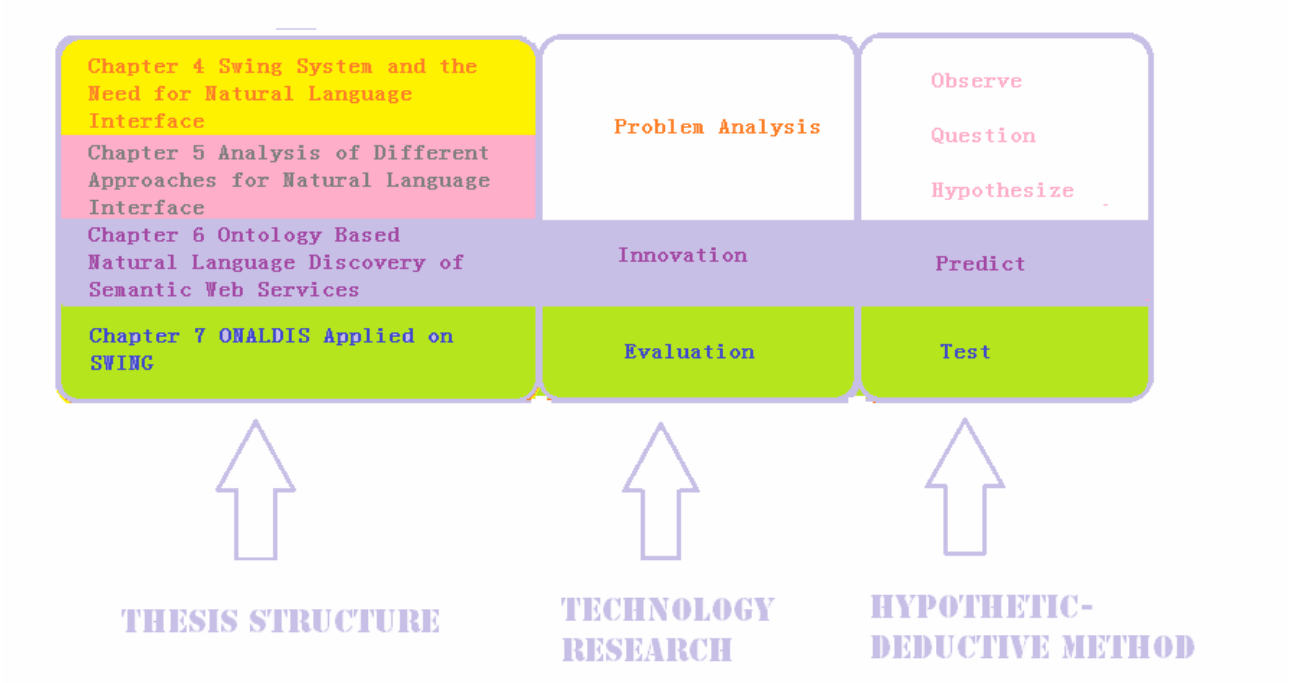


Figure 2-3 Research Methods and Structure

This chapter presented method of work employed within this thesis. They are Hypothetic-Deductive method and Technology research. How the methods are applied on thesis work is briefly discussed.

Chapter 3 Semantic Web Services and the Geospatial Domain

Semantic Web services and geospatial domain are two important concepts in this thesis. In order to be able to discovery semantic Web services, the construction of semantic Web services must be understood. This chapter briefly presents the current state of the relevant concepts/technologies including Semantic Web, Web services, and Semantic Web services.

3.1 Semantic Web

Semantic Web is different from original Web. In this section, the background of semantic Web is first introduced, and a definition of semantic Web is given. The general semantic Web structure is then presented. Some key technologies and standards in the structure are discussed. These are followed by summering some advantages by using semantic Web.

3.1.1 Background of Semantic Web

The World Wide Web is the largest single information resource humanity has ever produced. Unfortunately, despite its dependence on computers to operate at all, most of the information is only understandable by humans and not by computers. While computers can use the syntax of HTML documents to display them to you in a browser, Web computers can't understand the content—the semantics.

3.1.2 Understanding Semantic Web

Before discussing the term “Semantic Web”, let us take a look into the “normal Web” www. The World Wide Web is the largest single information resource humanity has ever produced. It is today is primarily based on documents written in HTML which is basically for visual presentation of context. The explicit meaning is not feasible for machines, which means that the machine (computer) can not understand the meaning of the context on Web. The main goal of Semantic Web is to have the machines “understand” the content of web elements.
Thus a definition of Semantic Web can be given below.

The **Semantic Web** is an evolving extension of the World Wide Web in which the semantics of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the Web content. [2][3]

Here is another definition of Semantic Web:

The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.

To help to understand Semantic Web concept, W3C [4] summarize Semantic Web’s essence as follows:

1) The Semantic Web is about two things. It is about common formats for integration and combination of data drawn from diverse sources, where on the original Web mainly concentrated on the interchange of documents. It is also about language for recording how the data relates to real world objects. That allows a person, or a machine, to start off in one database, and then move through an unending set of databases which are connected not by wires but by being about the same thing.

2) Semantic Web is a mesh of information linked up in such a way as to be easily processed by machines, on a global scale. You can think of it as being an efficient way of representing data on the World Wide Web, or as a globally linked database [5].

   Some structures should be followed when designing a semantic Web. A typical structure of Semantic Web is given in Figure 3-1.

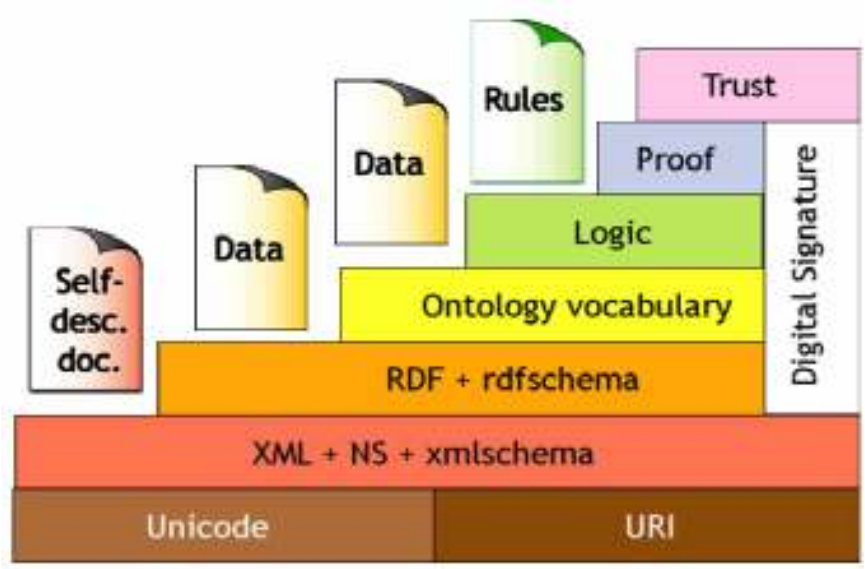


Figure 3-1 Semantic Web Stack [6]

The Semantic Web structure might be arranged in such way:
1) Each resource is identified with and Unified Resource Identifier (URI)
2) Each resource is described with a formal language (for example with RDF)
3) The relations among resources are described with the help of ontology so resources are linked to each other.
4) Construct a powerful logical language to make semantic Web expressive enough to help us in a wide range of situations
5) Contain proof checking mechanisms and trust dependent on context.

Here I give some examples which simply explained the first three concepts:

1. Use URI to identify a resource:

What is URI?
Wikipedia describes the concept URI as the following: [7]
URI means "Unified Resource Identifier" which is a compact string of characters used to identify or name a resource. The main purpose of this identification is to enable interaction with representations of the resource over a network, typically the World Wide Web.

Below is an URI which identifies an article.
http://folk.uio.no/yuhzuy/masterthesis/
Here "article" is a resource on Web. By using URI to identify it, any time we see this URI http://folk.uio.no/yuhzuy/masterthesis/, we know it points to this web resource.

2. Use RDF [8] to describe a resource:

Below is an RDF description of resource http://folk.uio.no/yuhzuy/masterthesis/ and http://ifi.uio.no/csinstitute/

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:foaf="http://xmlns.com/0.1/foaf/" >
      <rdf:Description rdf:about="http://folk.uio.no/yuhzuy/masterthesis/">
            <dc:creator rdf:parseType="Resource">
                  <foaf:name>Cathy Yang</foaf:name>
            </dc:creator>
            <dc:title>QA system and semantic web services</dc:title>
      </rdf:Description>
</rdf:RDF>
```

This piece of RDF code basically says that this article (http://folk.uio.no/yuhzuy/masterthesis/) has the title "QA system and semantic web services", and was written by someone whose name is "Cathy Yang".

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:foaf="http://xmlns.com/0.1/foaf/" >
      <rdf:Description rdf:about="http://ifi.uio.no/csinstitute/">
            <dc:name>Institute of Computer Science</dc:title>
            <dc:location>
                  <foaf:gateAddress> Gaustadalléen 23</foaf: gateAddress >
            </dc: location >
      </rdf:Description>
</rdf:RDF>
```

This piece of RDF code describes the resource (http://ifi.uio.no/csinstitute) which has name “Institute of Computer Science”, and location with gate address “Gaustadalléen 23”.

3. Use OWL [9] to describe the relations between two resource:

Below is an OWL description:

```
<owl:Class rdf:ID = “#articleX”>
            <isOwnedBy rdf:resource = “#InstituteOfComputerScience”>
</owl:Class>
```

This piece of OWL code describes the relation between “#articleX” and “#InstituteOfComputerScience”.

3.1.3 Semantic Web Technologies

There are several different technologies and standards related to term Semantic Web. In this section, they will be briefly presented.

According to [10], the basis for the functionality of the Semantic Web is the following:

1) A global naming scheme (URIs);
2) A standard syntax for describing data (RDF);
3) A standard means of describing the properties of that data (rdf-schema [11]);
4) A standard means of describing relationships between data items (ontologies);
5) The means to support trust and security.

In order to support these above, there are some main standards in the semantic Web area, which are listed as following:
1) RDF
2) RDFS (RDF Schema)
3) OWL
I will briefly talk about each of the standards:

RDF

RDF stands for **R**esource **D**escription **F**ramework. It is a framework for describing resources on the web. RDF provides a model for data, and syntax so that independent parties can exchange and use it. It is designed to be read and understood by computers, not for being displayed to people.

RDF Schema

RDFS or RDF Schema is an extensible knowledge representation language, providing basic elements for the description of ontologies, otherwise called RDF vocabularies, intended to structure RDF resources.

OWL

The **W**eb **O**ntology **L**anguage OWL is a semantic markup language for publishing and sharing ontologies on the World Wide Web. OWL is developed as a vocabulary extension of RDF (the Resource Description Framework) and is derived from the DAML+OIL Web Ontology Language.

3.1.4 Advantages of Semantic Web

Paper "POTENTIAL ADVANTAGES OF SEMANTIC WEB FOR INTERNET COMMERCE" discusses 12 advantages of semantic Web [12] which are listed as the following:

1) Search
2) Agents
3) Knowledge management (KM)
4) Integration
5) Composition of complex systems
6) Multimedia collection
7) Information filtering
8) Machine dialogue across the domains
9) Virtual community
10) Online advertising
11) Serendipity (unexpected benefits)
12) Vocabulary flexibility & standardization

3.2 Web Services

As mentioned before, Web resources can be in somehow divided into two parts, data and services, where data means the collection of Web resources and Web Services stand for those services that are available over the Internet, use a standardized XML messaging system, and are not tied to any

one operating system or programming language. Nowadays the term Web services has gained more and more Web users/programmers' focus. In this section, a brief description of Web services is given first. The general Web service structure is then presented. Some key technologies and standards in the structure are then discussed.

3.2.1 Understanding Web Services

Web Service [13] is Extensible markup Language (XML) applications mapped to programs, objects, or databases or to comprehensive business functions. Using an XML document created in the form of a message, a program sends a request to a Web service across the network, and, optionally, receives a reply, also in the form of an SML document. Web services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services.

The W3C Web service definition encompasses many different systems, but in common usage the term refers to clients and servers that communicate using XML messages that follow the SOAP standard. Common in both the field and the terminology is the assumption that there is also a machine readable description of the operations supported by the server written in the Web Services Description Language (WSDL). The latter is not a requirement of a SOAP endpoint, but it is a prerequisite for automated client-side code generation in many Java and .NET SOAP frameworks (frameworks such as Spring and Apache CXF being notable exceptions). Some industry organizations, such as the WS-I, mandate both SOAP and WSDL in their definition of a Web service.

Figure 3-2 shows Web Service architecture.

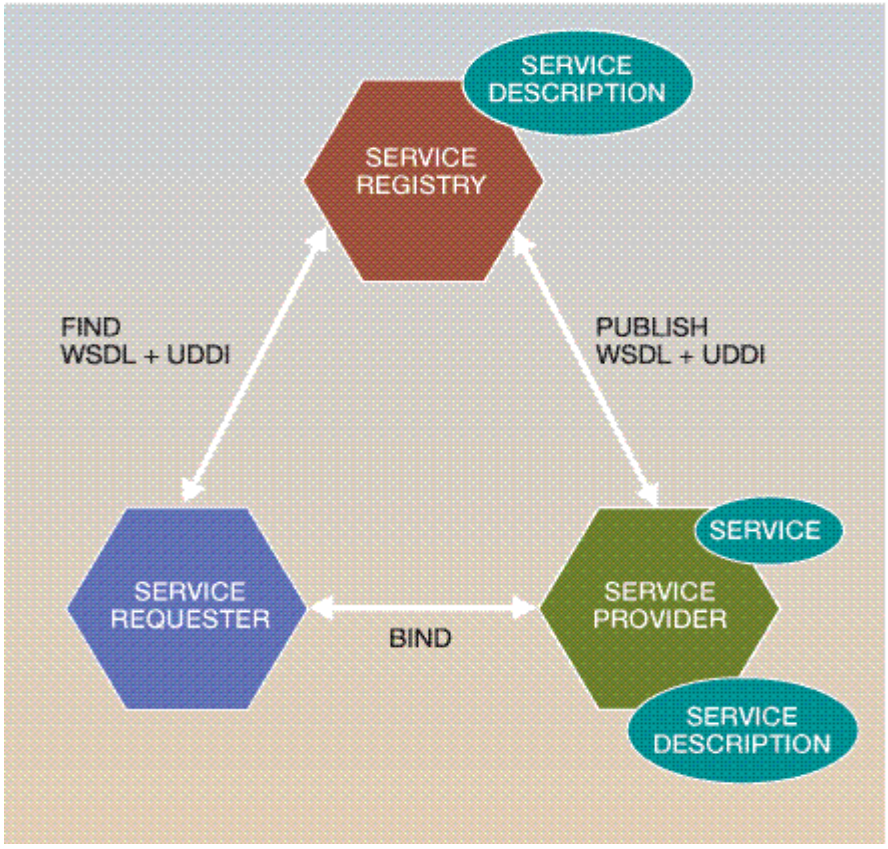


Figure 3-2 Web service architecture [14]

Figure 3-3 shows the major Web service roles and how they interact with each other. Service provider is the provider of the Web services. It implements the service and makes it available on the Internet. Services requestor is the consumer of the Web service. It utilizes an existing Web service by opening a network connection and sending an XML request. Service registry is logically centralized directory of services. It provides a central place where developers can publish new services or find existing ones.

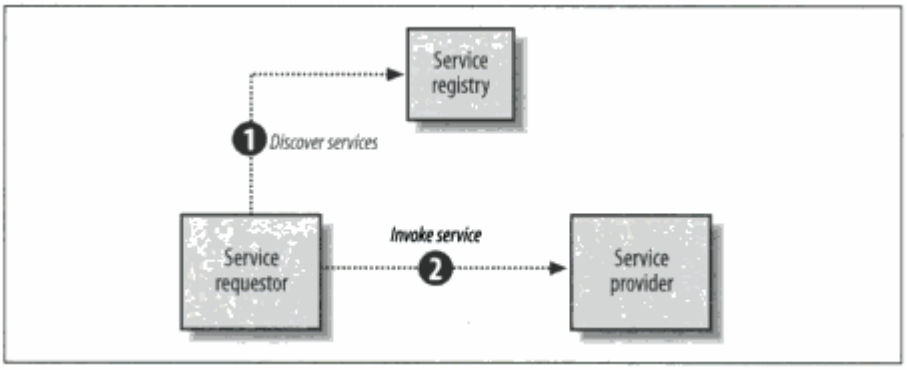


Figure 3-3 Web Service roles [15]

Figure 3-4 shows Web Service structure.

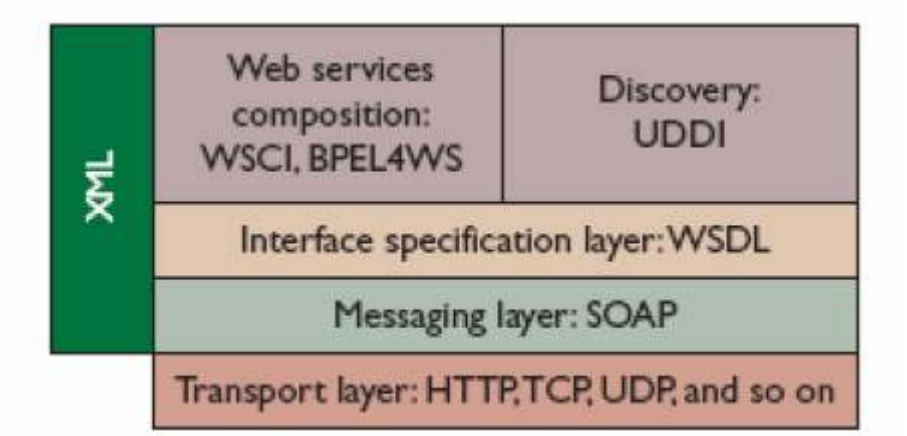


Figure 3-4 Web service Structure

Another variant is showed in Figure 3-5 Web Service Protocol Stack .

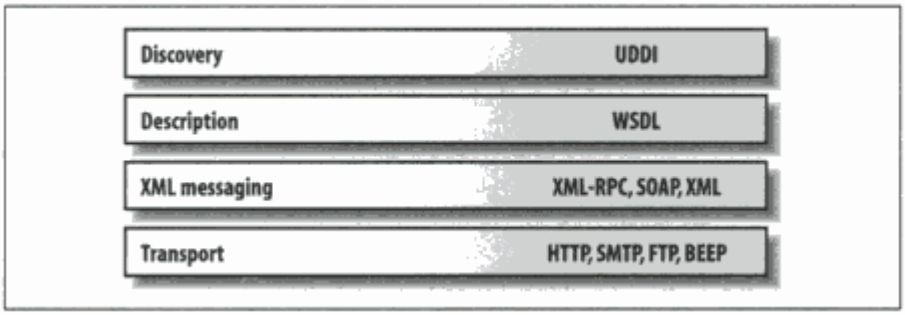


Figure 3-5 Web Service Protocol Stack [15]

This Web service protocol stack is still evolving, but currently there are four main layers. Following is a brief description of each layer.

1) Service transport
   This layer is responsible for transporting messages between applications. Currently, this layer includes hypertext transfer protocol (HTTP), Simple Mail Transfer Protocol (SMTP), file transfer protocol (FTP), and newer protocols, such as Blocks Extensible Exchange Protocol (BEEP).
2) XML messaging
   This layer is responsible for encoding messages in a common XML format so that messages can be understood at either end. Currently, this layer includes XML-RPC and SOAP.
3) Service description
   This layer is responsible for describing the public interface to a specific Web service. Currently, service description is handled via the Web Service Description Language (WSDL).
4) Service discovery
   This layer is responsible for centralizing services into a common registry, and providing easy publish/find functionality. Currently, service discovery is handled via Universal Description, Discovery, and Integration (UDDI).

3.2.2 Web Service Web Technologies and Standards

Web services standards define the format of the message, specify the interface to which a message is sent, describe conventions for mapping the contents of the message into and out of the programs implementing the service, and define mechanisms to publish and to discover Web services interfaces. Some Web services standards/technologies are presented as following:

1) UDDI [16]

UDDI is a service registry which provides a Web services directory platform. It is a centralized Web service search engine helping Web consumer applications to find adequate service offerings. In a UDDI registry, the following may be found. a) Information about businesses and organizations offering Web services. b) Descriptions of the Web services that these organizations provide. c) Information about technical interfaces to these Web services.

2) WSDL [17]

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. It was developed by Microsoft, Ariba, and IBM. It establishes a common format for describing and publishing Web service information.

3) WSCI [18] (OASIS)

WSCI was written by BEA, Intalio, Sun, and SAP in 2002. Its syntax and semantics are strikingly similar to those of BPML. It describes the dynamic interface of the Web service participating in a given message exchange by means of reusing the operations defined for a static interface. This is expressed in terms of temporal and logical dependencies among the exchanged messages, featuring sequencing rules, correlation, exception handling, and transactions. WSCI also describes the collective message exchange among interacting Web services, thus providing a global, message-oriented view of the interactions. WSCI works in conjunction with the Web Service Description Language (WSDL), the basis for the W3C Web Services Description Working Group. It can also work with another service definition language that exhibits the same characteristics as WSDL. The specification is being brought to the attention of relevant W3C working groups, including the Web

Services Architecture Working Group, the Web Services Description Working Group, and the Web Ontology Working Group.

4) BPEL [20]
BPEL stands for Business Process Execution Language. It is one of the most popular languages for Web service composition. It is an XML-based language designed to enable task-sharing for a distributed computing or grid computing environment - even across multiple organizations - using a combination of Web services. Written by developers from BEA Systems, IBM, and Microsoft, BPEL combines and replaces IBM's Web Services Flow Language (WSFL) and Microsoft's XLANG specification. (BPEL is also sometimes identified as BPELWS or BPEL4WS.)

5) SOAP [21]\
SOAP is a simple XML-based protocol to let applications exchange information over HTTP. Today's applications communicate using Remote Procedure Calls (RPC) between objects like DCOM and CORBA, but HTTP was not designed for this. RPC represents a compatibility and security problem; firewalls and proxy servers will normally block this kind of traffic. A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this. SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

3.3 Semantic Web Services

In this section, the background of semantic Web services is introduced first. The definition and the general Web service structure are then presented. Some key technologies and standards in the structure are discussed, and the advantages/features are discussed at the end.

3.3.1 Background of Semantic Web Services

In the last section, Web services are presented. Normally Web services are expected to be integrated as part of Web processes. There is a growing consensus that Web services alone will not be sufficient to develop valuable Web processes due to the degree of heterogeneity, autonomy, and distribution of the Web. It is agreed that it is essential for Web services to be machine understandable in order to support all the phases of the lifecycle of Web processes. That is the reason why Web services are considered to be associated with semantic technologies.

3.3.2 Understanding Semantic Web Services

Semantic Web Services is an application of Semantic Web. It combines Semantic Web and Web Services. Ontologies are used to make the Web Services understandable by machine, and thus enable automatic discovery, invocation and composition of Web Services.

Figure 3-6 shows the Semantic Web Service structure

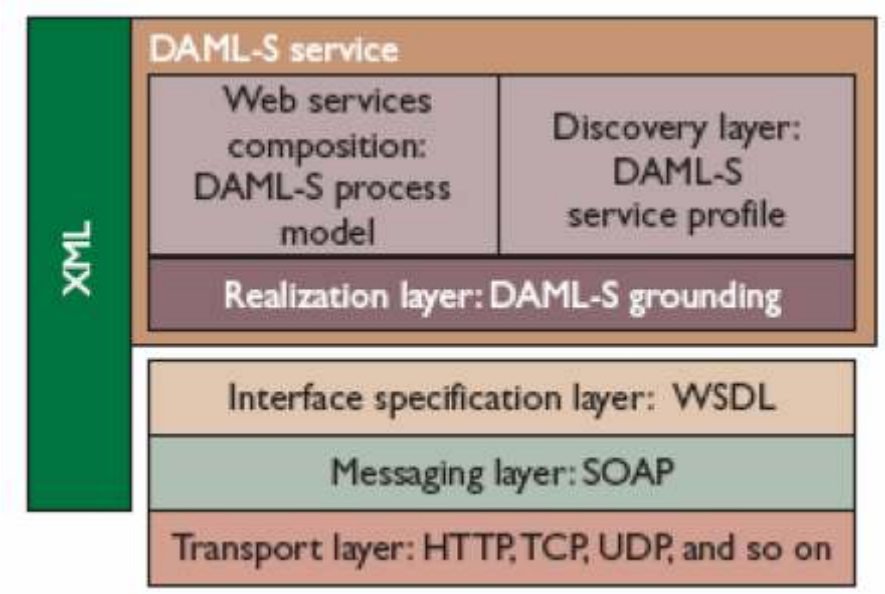


Figure 3-6 Semantic Web Service Structure

Comparing Figure 3-4 Web service Structure and Figure 3-6 Semantic Web Service Structure, as shown in Figure 3-7, it is easy to tell that the differences between them are the top layers. Both WS and SWS have "Web services composition" component and "Discovery" component, where the technologies for implementing the components are slightly different. Besides, SWS has a "Realization layer" and discovery layer, composition layer and realization layer are wrapped together to form a "service". More details can be found in paper collection [22].

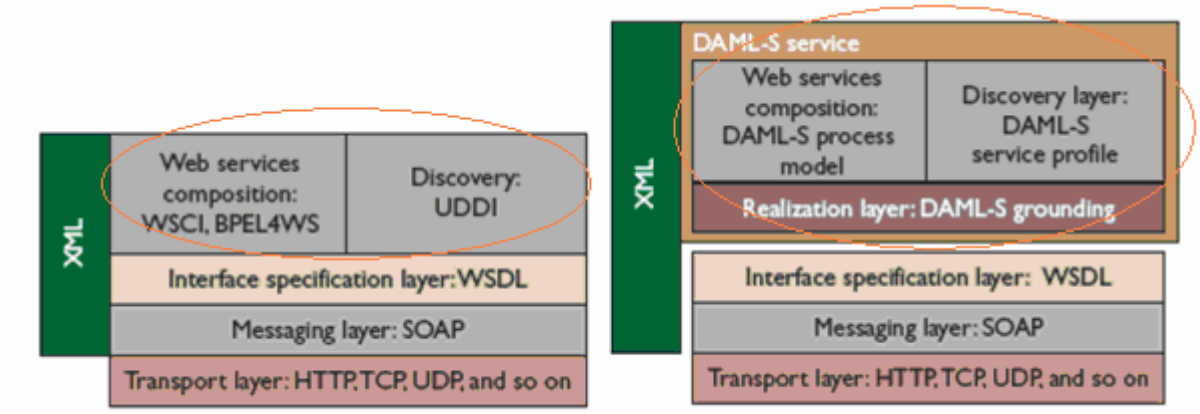


Figure 3-7 Difference of WS and SWS structure

Figure 3-8 illustrates an example of the semantic Web services integration platform in semantic Web scenario. A **service provider** describes the services it offers. This service description is expressed in terms of an **ontology** and is published in a **Service Registry** for users to find. It is described through **semantics** which is a part of the Semantic web. The description includes this semantics and is added as an addition to the Semantic Web or from already described domains placed in the Semantic Web. The **service requestor** creates a service request describing the service it needs which also includes **semantic description** of what service is needed. The request of the Service requestor is sent to a **service discovery engine** that locates the service wanted. In the **Web service integration platform** there is in addition to the registry and the discovery services a **Service Invocation engine** that communicates with the requestor's client and the provider's server.

The Web service Integration Platform understands and uses the ontologies in the semantic Web to fulfill the requests from the requestor.

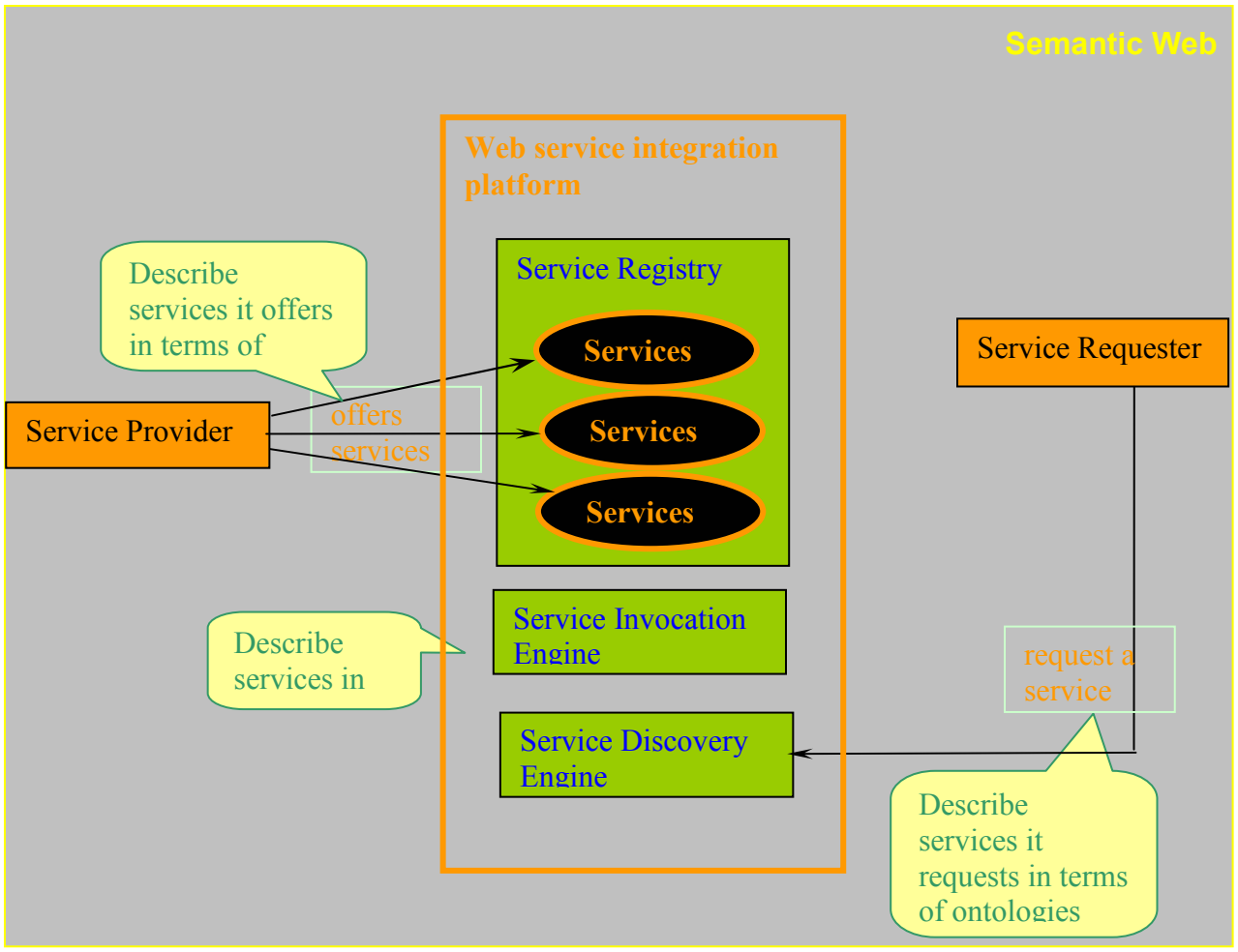


Figure 3-8 Semantic Web Service

3.3.3 Some Standards of Semantic Web Services

W3C Semantic Web Services Interest group provides an open forum to discuss Semantic Web Services. Submissions by W3C members include [23]:

1) **OWL-S (Web Ontology Language for Services)**
2) **WSMO (Web Service Modeling Ontology)**
3) **WSDL-S (Web Service Semantics)**

3.3.3.1 OWL-S (Web Ontology Language for Services) [24]

To make use of a Web service, a software agent needs a computer-interpretable description of the service, and the means by which it is accessed. An important goal for Semantic Web markup languages, then, is to establish a framework within which these descriptions are made and shared. Web sites should be able to employ a standard ontology, consisting of a set of basic classes and properties, for declaring and describing services, and the ontology structuring mechanisms of OWL provide an appropriate, Web-compatible representation language framework within which to do this. **OWL-S** is such ontology which originates from the DAML (DARPA Agent Markup Language), and is build on top of OWL (Web Ontology Language). The service ontology aims supporting

discovery, composition and invocation of semantic Web Services by adding semantic descriptions.

Tools related to OWL-S:
1) OWL-S Protégé-based Editor [25]
2) OWL-S matcher [26]
3) OWL-S plug-in for Axis
More details about OWL-S can be read from [28].

3.3.3.2 WSMO (Web Service Modeling Ontology) and WSML [29]

The mission of **WSMO** is to describe various aspects of Semantic Web Services, and to solve the integration problem. It aims to provide a world-wide standard, which will be developed together with industrial partners and other research groups, and will be aligned with many different research projects through its main features: simplicity (a solution to the integration problem that is as simple as possible), completeness (solves all aspects of the integration problem), excitability (a set of execution semantics exists as well as a reference implementation).

Tools related to WSMO:
1) WSMO Studio [30]
2) Protégé with WSMO Tab plug-in [30]
3) WSML Rule Reasoner [31]
4) WSMO4J [32]
More details about WSMO can be read from [33].

WSML provides a formal syntax and semantics for WSMO to describe the elements defined in WSMO. More details about WSML can be read from [33].

3.3.3.3 WSDL-S (Web Service Semantics) [33]

WSDL-S builds on the WSDL standard. It is an extension of the syntactical level of WSDL, and includes semantic capabilities for Semantic Web Services. In WSDL-S we assume that an ontology model already exists and they are maintained outside of WSDL documents and are only referred to through extensibility elements.

More details about WSDL-S can be read from [33].

3.3.4 The Features of Semantic Web Services

What are the features of semantic Web services comparing to Web services? According to W3C's documentation about OWL-S, the Semantic Web should enable users to locate, select, employ, compose, and monitor Web-based services automatically. The following are the details.

- **Automatic Web service discovery (with OWL-S as an example)**

With OWL-S mark-up of services, the information necessary for Web service discovery could be specified as computer-interpretable semantic mark-up at the service Web sites, and a service registry or ontology-enhanced search engine could be used to locate the services automatically. Alternatively, a server could proactively advertise itself in OWL-S with a service registry, also called middle agent, so that requesters can find it when they query the registry. Thus, OWL-S provides declarative advertisements of service properties and capabilities that can be used for

automatic service discovery. Illustration (Figure 3-9) is followed.

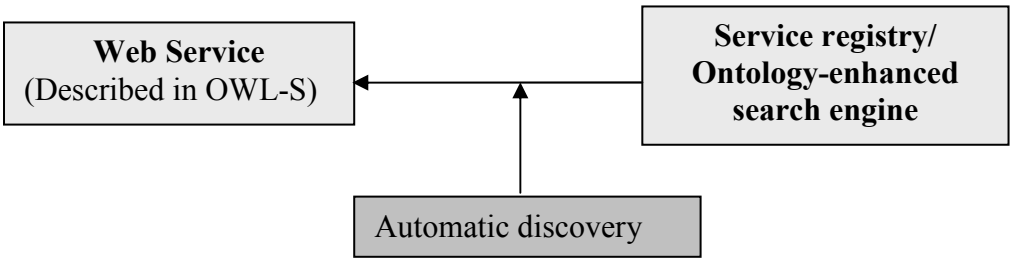


Figure 3-9 Automatic Web Service Discovery

- **Automatic Web service invocation**

Automatic Web service invocation is the automatic invocation of a Web service by a computer program or agent, given only a declarative description of that service, as opposed to when the agent has been pre-programmed to be able to call that particular service. Execution of a Web service can be thought of as a collection of remote procedure calls. OWL-S mark-up of Web services provides a declarative, computer-interpretable API that includes the semantics of the arguments to be specified when executing these calls, and the semantics of that is returned in messages when the services succeed or fail. A software agent should be able to interpret this mark-up to understand what input is necessary to invoke the service, and what information will be returned. OWL-S, in conjunction with domain Ontologies specified in OWL, provides standard means of specifying declaratively APIs for Web services that enable this kind of automated Web service execution. Illustration (Figure 3-10) is followed.

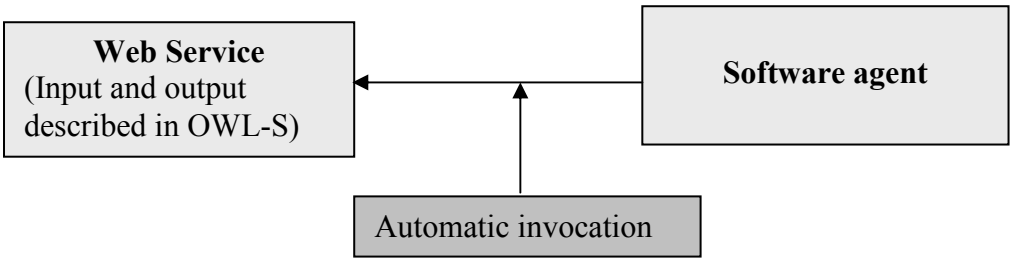


Figure 3-10 Automatic Web Service Invocation

- **Automatic Web service composition and interoperation**

With OWL-S mark-up of Web services, the information necessary to select and compose services will be encoded at the service Web sites. Software can be written to manipulate these representations, together with a specification of the objectives of the task, to achieve the task automatically. To support this, OWL-S provides declarative specifications of the prerequisites and consequences of application of individual services, and a language for describing service compositions and data flow interactions.

3.3.5 Related work in Web Services composition

In this section, some works in Web Service composition field are summarized

SWORD [30] is a model for web service composition. It uses its own simple description language and does not support any existing standards like WSDL or OWL-S. Services are modeled using inputs and outputs, which are specified using an Entity Relationship model. Inputs are classified into conditional inputs and data inputs. Outputs are classified similarly. Conditional inputs/outputs are assertions about entities on which the service operates and the relationships between entities. Data inputs/outputs constitute the actual data (attributes of entities) that the service uses.

A similar framework was developed at **IBM Research Laboratories** as part of the Web Services Toolkit (WSTK) [31]. A composition engine [32] has been built for services described in WSDL. This work describes the use of a planner for composition, constructing operators from the service description is not fully automated. This is primarily because of the absence of a mechanism to capture domain knowledge in WSDL.

The **Golog-based system** [33] does service composition by using general templates that are modified based on user preferences, yielding the final plan. The templates are not automatically built and constitute part of the plan.

Semantic E-Workflow Composition [34] talks about composition in workflow systems. Workflow is an abstract representation of a process. A workflow is built using components called tasks/activities. Traditionally, appropriate tasks are selected from a workflow repository. This work introduces the notion of using web services as tasks in the workflow. They address the issue of selecting appropriate web services using semantic discovery. They also discuss how web services can be integrated into workflows by syntactic and semantic integration of inputs and outputs. The primary contribution of this work is to provide a tool to assist in manual workflow composition. This work does not deal with OWL-S based semantic web services.

SHOP2 is a Hierarchical Task Network (HTN)-based planner for composing web services [35]. This is first to deal with composition using DAML-S/OWL-S. The SHOP2-based composer requires that each atomic service either produces outputs or effects but not both. This assumption has been made to differentiate between information-gathering services and effect-producing services. By making this differentiation, SHOP2 executes the information-gathering services during plan generation and simulates the execution of services that produce effects. To use SHOP2 for composition, service providers have to describe services that are consistent with the above assumption. In principle, services can be re-designed so that each service either produces only outputs or only effects but not both.

Paolucci, Sycara and Kawamura in their work [36] make a reference to **RETSINA**, a planner that makes use of the HTN planning paradigm. This planner is similar to SHOP2 in the way they interleave planning and execution. They claim that by executing the information-gathering services during planning, unexpected situations can be handled by re-planning. They make use of OWL-S in their work but do not mention how the RETSINA planner has been modified to use OWL-S descriptions.

3.3.6 Frameworks in Semantic Web Service Discovery, Composition and Invocation

In this section, two frameworks which can discover, compose and invoke semantic Web Services are shortly introduced.
The first one is SAP's GP Framework(Figure 3-11), adapted from [44]

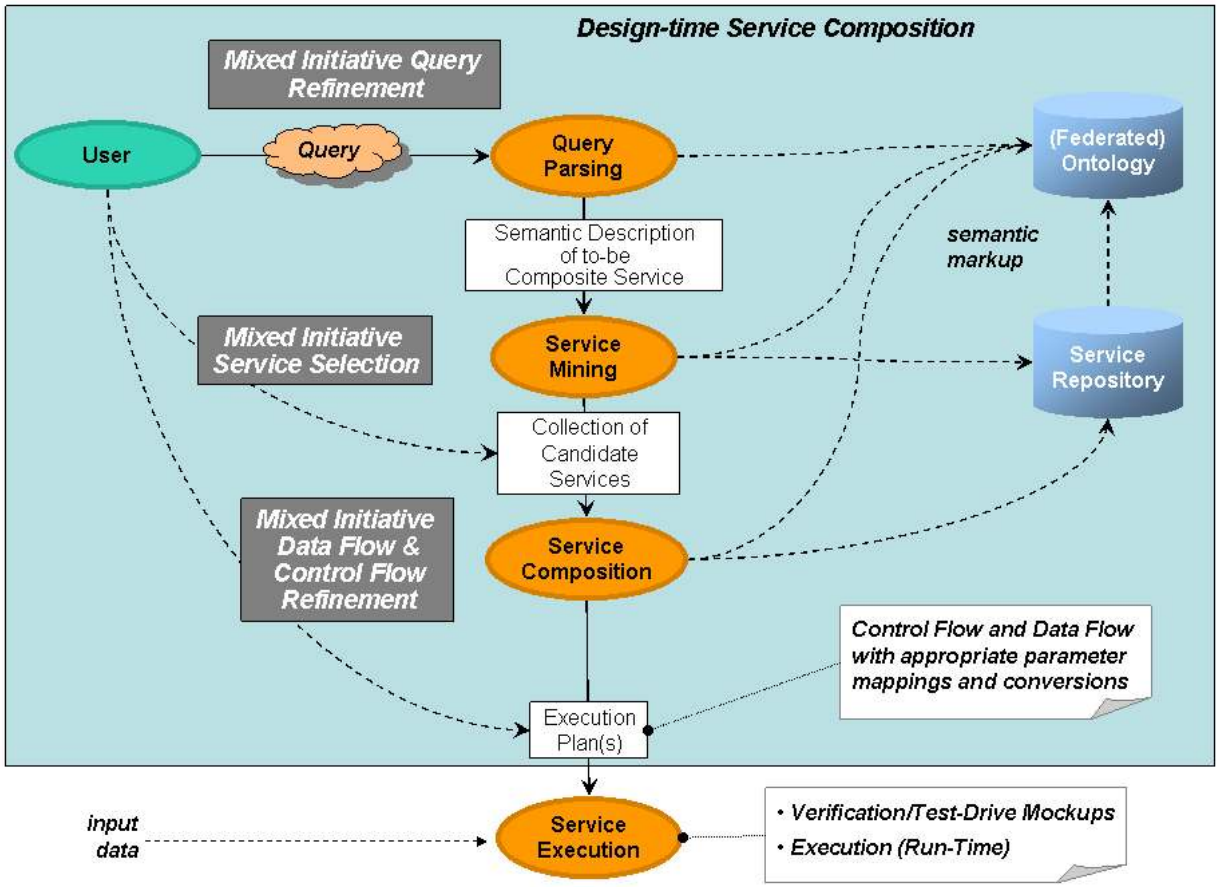


Figure 3-11 SAP's GP Framework - Simplified Workflow

With this framework, user can achieve query parsing, service mining, and service composition, based on Ontology.

The second one is the SWING framework (Figure 3-12), adapted from [43].

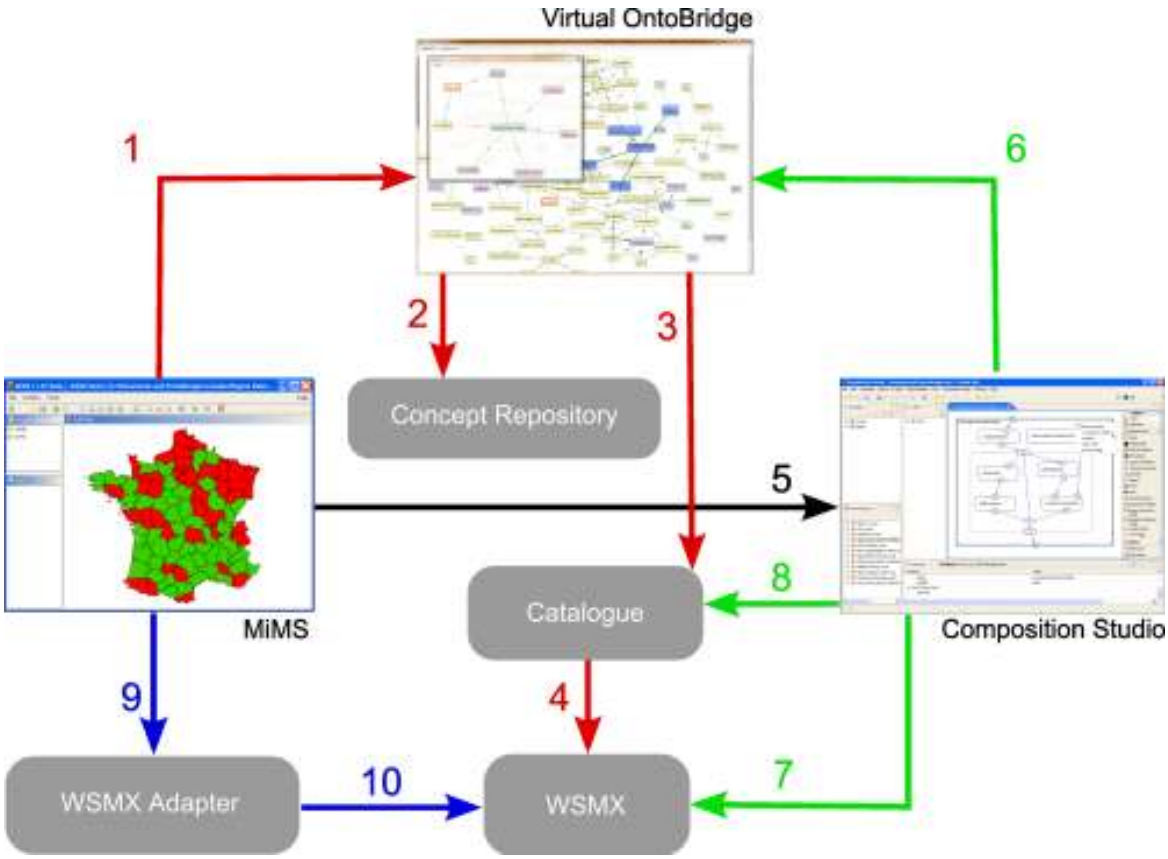


Figure 3-12 SWING Framework

With the SWING framework, user can achieve semantic Web services discovery, semantic Web services execution, and semantic Web services annotation.

3.4 Geospatial Domain

In this thesis the problem that will be addressed is in the geospatial domain. This section introduces basic concepts in the geospatial domain.

3.4.1 Geospatial Information

Geospatial information is information describing the location and names of features beneath, on or above the earth's surface. At its simplest this can mean the basic topographical information found on a map, but also includes different location-related datasets combined into complex layers that show information such as land use and population density.

3.4.2 Geospatial Data

Geospatial data on the Web contains more and more important information. Capturing, analyzing and managing these data can help people to achieve various kinds of tasks. These Geographical data is distributing on the Web in an irregular way which makes data capturing and analyzing process not that trivial. There exist many GIS applications which are widely used for scientific investigations, resource management, asset management, archaeology, environmental impact assessment, urbanplanning, cartography, criminology, geographichistory, marketing, logistics, prospectivity mapping, and other purposes. For example, GIS might allow emergency planners to

easily calculate emergency response times in the event of a natural, GIS might be used to find wetlands that need protection from pollution, or GIS can be used by a company to site a new business location to take advantage of a previously under-served market.

Chapter 4 SWING System and a Natural Language Interface

This chapter is about the SWING system and the need for a natural language interface. The background of SWING (Semantic Web services INteroperability for Geospatial decision making) [43] project is first presented, and then the structure and the components of SWING system are introduced. Based on the understanding of SWING system and its current interface, the need for a natural language interface is suggested.

4.1 SWING Project

SWING is a project of the Information Society Technologies (IST) Program for Research, Technology Development & Demonstration under the Sixth Framework Program of the European Commission, which was active from April 2006 to April 2009. The objective of SWING was to provide an open, easy-to-use SWS framework of suitable ontologies and inference tools for annotation, discovery, composition, and invocation of geospatial web services.

4.1.1 The Background

Today, a number of non-semantic web services are available within the geospatial domain. The scarcity of semantic annotation and the lack of a supportive environment for discovery and retrieval make it difficult to employ such services to solve a specific task in geospatial decision making. And a comprehensive knowledge of logics, ontologies, metadata and various specification languages is required to describe a service semantically. Therefore SWING aims at deploying Semantic Web Service (SWS) technology in the geospatial domain. In particular, the project addresses two major obstacles that must be overcome for SWS technology to be generally adopted, i.e. to reduce the complexity of creating semantic descriptions and to increase the number of semantically described services.

4.1.2 Structure

The objective of the SWING project is to create a Semantic Web Service (SWS) framework for the geospatial domain. This framework will consist of a set of tools and ontologies, which will simplify the access to SWS technology and in this way, reduce the training required for applying this technology. The tools will allow a user to perform basic tasks such as annotation, composition, discovery, and execution of web services.
In this section, adapted from [43], the SWING framework is divided into several modules. Each module is generally introduced in the following.

4.1.2.1 Geospatial decision-making application

BRGM will implement the geospatial decision-making application. MiMS (**Mi**neral resources **M**anagement **S**ystem) will be a prototype used by domain experts to create and publish documents (based on dynamic maps) for decision support in mineral resource management. MiMS, as the end-user application, relies on the others components.

4.1.2.2 Semantic Discovery and Execution Engine

The Web Service Modeling eXecution environment (WSMX) will provide the Semantic Web Service infrastructure for the project. It will consist of tools for discovery, execution, mediation,

and orchestration of web services. WSMX applies the Web Service Modelling Language (WSML) as its formal syntax. WSML has four main elements:

- Ontologies, which provide a formalization of the domain.
- Web services, which provide some functionality which can be requested.
- Goals, which define some objective that a client may request from a web service.
- Mediators, which handle interoperability problems between the other components such as differences in semantics.

WSMX will be used by the Development Environment for discovery and execution of web services and web service compositions.

4.1.2.3 Geospatial Ontologies

The University of Munster (UoM) will develop a set of ontologies for the SWING platform. These will be used for solving interoperability problems between web services and for discovery and annotation of web services. The ontologies will be made available through an ontology repository to the other SWING tools.
The Service annotation tool and the Development Environment will use these ontologies in their annotation process.

4.1.2.4 Semantic Annotation Engine

The semantic annotation engine will be implemented by the Josef Stefan Institute (JSI). It will apply text mining and machine learning techniques to create semantic annotations for web services. The idea is to use the annotation engine to increase the number of semantically annotated web services.
The Development Environment will use the service annotation engine for automatic annotation of web service compositions.

4.1.2.5 Service Catalogue

IONIC will provide the service catalogue. It is a standard OGC catalogue service, which provides information about OGC web services. The catalogue will be enhanced with the ability to store and discover semantically annotated services. IONIC is also responsible for providing the web services needed for the project.

4.1.2.6 Development Environment

The Development Environment (DEV) will be developed by SINTEF. It will provide support for constructing web services compositions and allow a user to annotate and discover web services using semantic information.
Moreover, DEV will provide facilities for executing and deploying compositions. In order to achieve its task the Development Environment will use the work done in the other work packages as well as build upon work done by SINTEF in other projects. The Visual Service Composition Studio has been chosen as the basis for the Development Environment because it is open source, developed by SINTEF and operates on an abstract UML level when composing web services.

4.1.3 The components in the SWING Architecture

In this section adapted from [43] the authors describes the components in SWING system as Figure 3-12, and the connections between them. Each component is briefly introduced in the following

sections.

4.1.3.1 MiMS

MiMS (**Mi**neral resources **M**anagement **S**ystem) is the Application prototype. This prototype will show dynamic discovery, composition and invocation of geospatial web services within the domain of sustainable exploitation of natural resources. Parts of the application prototype will be developed using the Development Environment. At run-time, the application prototype will utilise the Semantic Discovery and Execution component.

4.1.3.2 WSMX

WSMX is the Semantic Discovery and Execution component, which provides the basic Semantic Web Service infrastructure. It will be based on the WSMX technology developed in the DIP project, adapted to the geospatial domain. It consists of a repository of semantically described web services and a set of inference tools for discovery and dynamic invocation of the services.

4.1.3.3 ONTO

ONTO is Geospatial Ontology component, including an ontology maintenance strategy which ensures consistent and high-quality ontologies for all SWING components. The ontologies are central for semi-automatic annotation, semantic discovery, and composition of geospatial web services within the domain of natural-resource decision-support tasks.

4.1.3.4 ANNOT

ANNOT is the Annotation Engine which utilizes knowledge discovery techniques to aid the user in the annotation task. The annotation task can be seen as the process of establishing explicit relations between the domain ontology and a Web service schema. Semantically annotated Web services can be registered in the Catalog and in WSMX (WP2 & WP5) to support semantic discovery and execution.

4.1.3.5 CAT

CAT provides a standard web service registry interface storing entries to classical geospatial and non-spatial services. In addition, it utilizes the underlying components to provide semantically enhanced discovery functionality.

4.1.3.6 DEV

DEV is the Development Environment component, which is based on IBM's Open Source Eclipse Development Environment. It consists of a number of plug-ins that integrates and hides the complexity of the other components. Application and service developers can use the Development Environment to discover both semantic and non-semantic services, to semantically describe their services and to compose multiple services.

4.2 Ontologies Used in SWING

In SWING two core “types” of ontologies have been identified along with the development of specific strategies on how to acquire and formalize the knowledge needed for generating them. The two types of ontologies are Domain ontologies and Geospatial domain ontologies. Domain

Ontologies capture the specific view of an information community independent of how the information is encoded. A knowledge acquisition strategy that manages the interplay between ontology engineer and domain expert has been developed and tested. Geospatial Domain Ontologies capture the specifications for geodata encoding and processing. As the Open Geospatial Consortium (OGC) standards are used in SWING project, the geospatial ontologies on respective OGC specifications are mainly based, but the development of new ontologies is also supported. The future challenges to deal with are that ontology development is a time-consuming process and it should be supported with more interactive tools, including for instance handling of user feedback.

4.3 SWING Interface for Service Searching and Registration

Figure 4-1 is a screen shot of the MiMS application. It shows the interface for service searching in the SWING system.

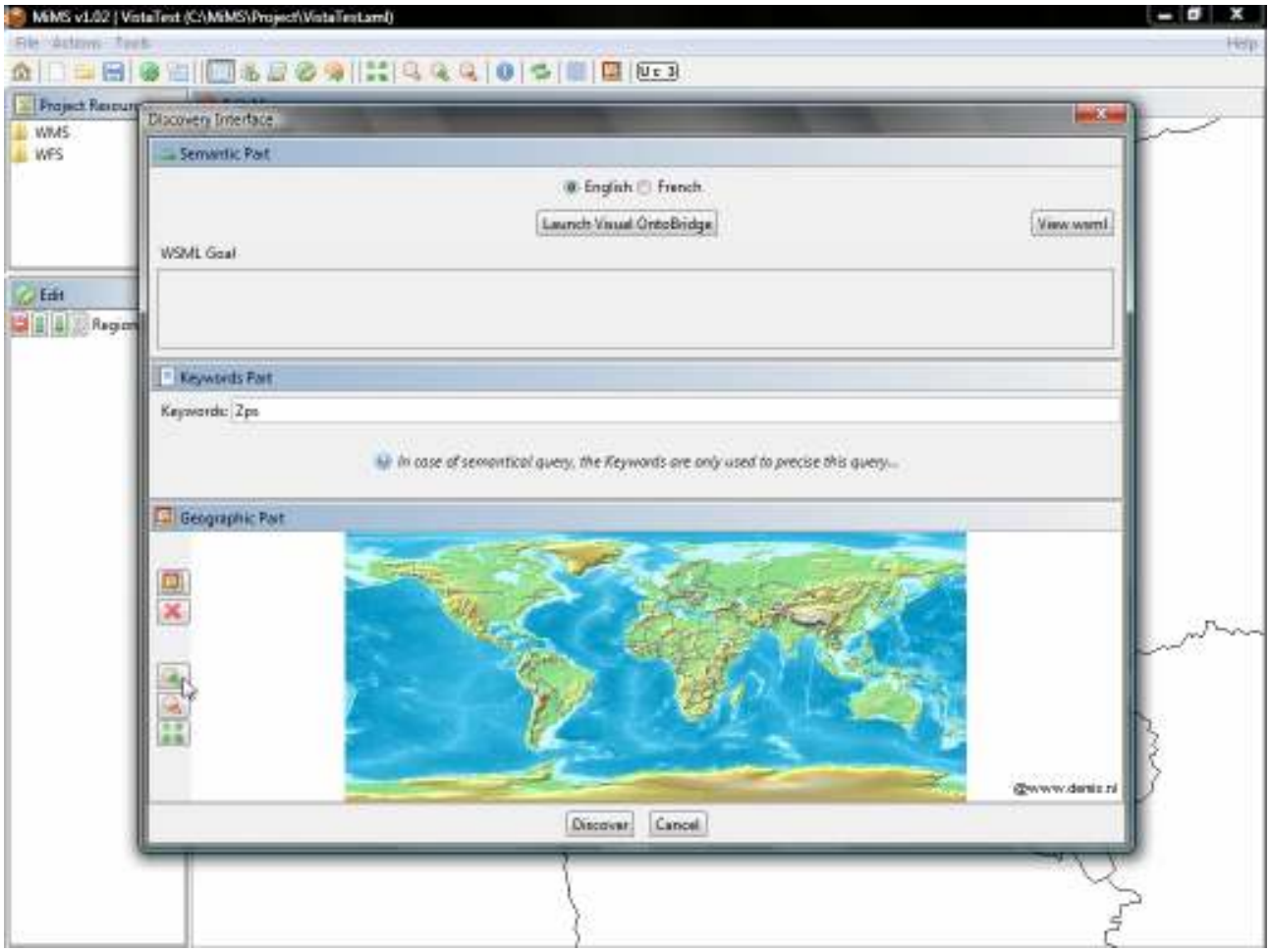


Figure 4-1 MiMS application

In the SWING system, there are basically two types of interfaces for Web services discovery and registration:

1. Keyword search + bounding box
2. Semantic ontology based search (OntoBridge) + bounding box

With the first approach, Keyword search + bounding box (concept “Bounding Box” here is an

action that is MiMS application user provides the geographical constraints), domain experts are supposed to give WSML goal (Figure 4-3), keyword and mark the bounding box on the map, which are three separated parts. With the second approach, Semantic ontology based search (OntoBridge) + bounding box, an external software module "OntoBridge" (Figure 4-2) is invoked to analyze semantic annotations, and generate queries from the semantic annotations. The user has to choose the concepts from the very complex ontology diagram (concept repository). In addition, the bounding box also needs to be defined.

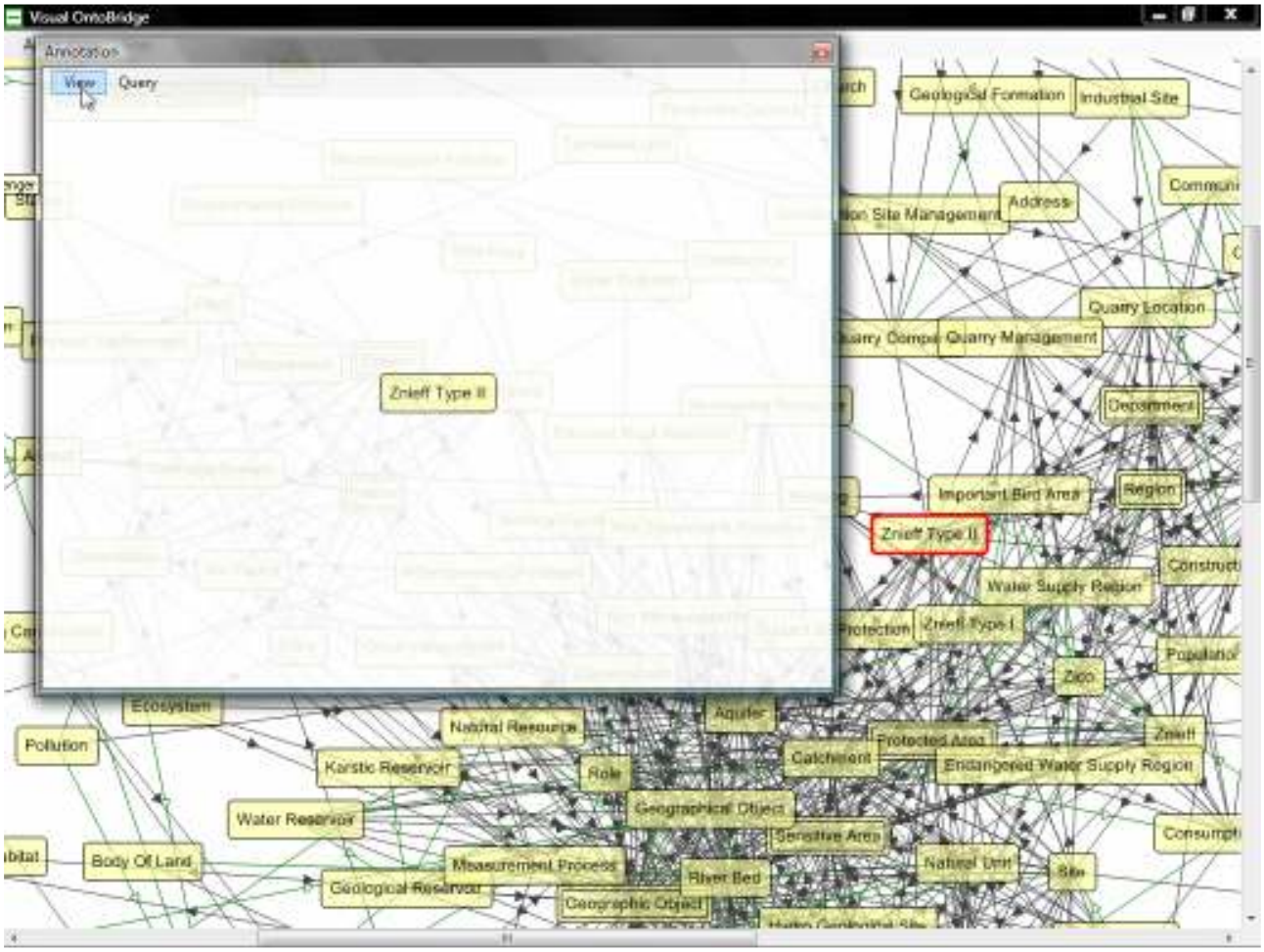


Figure 4-2 OntoBridge

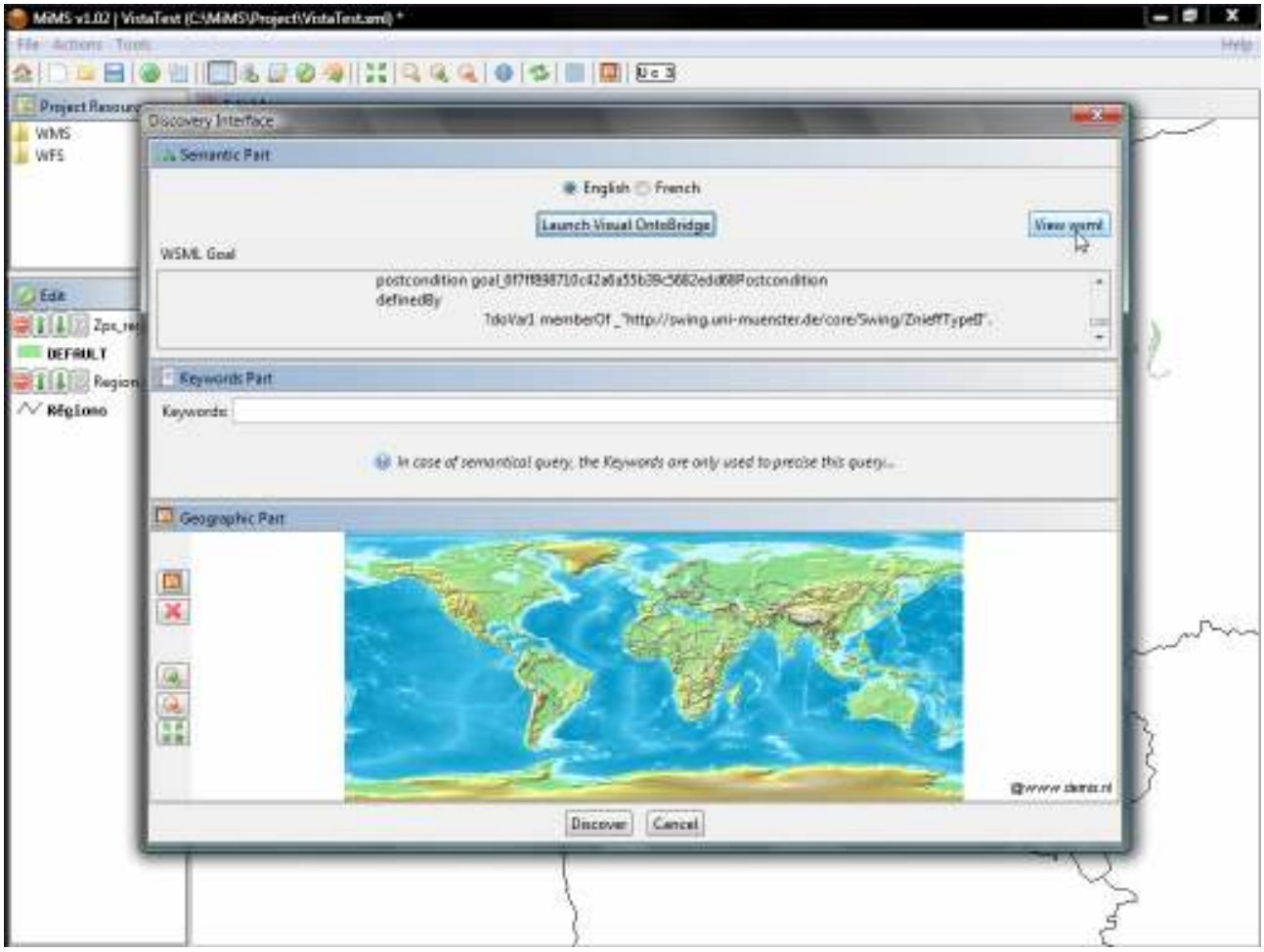


Figure 4-3 MiMS WSML Goal

4.3.1 Challenges with the current interface

The challenge with the current interface is that the underlying concepts of MiMS are of rather high technological level for end-users, as they need to have a basic understanding of the discovery and use of semantic web services New skills are required for end-users to master the system

The challenge with using bounding box is that end users have to know roughly where the location is on the map. For instance, the user wants to find the Web services related to the zinc mine in Nepal, he/she has to know where Nepal is in order to user the bounding box.

With the natural language interface, the users can directly enter a natural language question as input, the system will be able to retrieve the keywords and relevant geo info (instead of the bounding box) from the natural language queries and generate the WSML goal automatically.

4.3.2 Natural Language Interface for Feature Type Annotation

Natural language search interface is also used in SWING for annotating feature type.
The interface below, Figure 4-4 shows that when annotating a feature “Zico”, instead of looking for the concept from the complex ontology graph, SWING provides the natural language query search where user can give a natural language query and get back a list of proposed concepts and a list of

proposed triples.

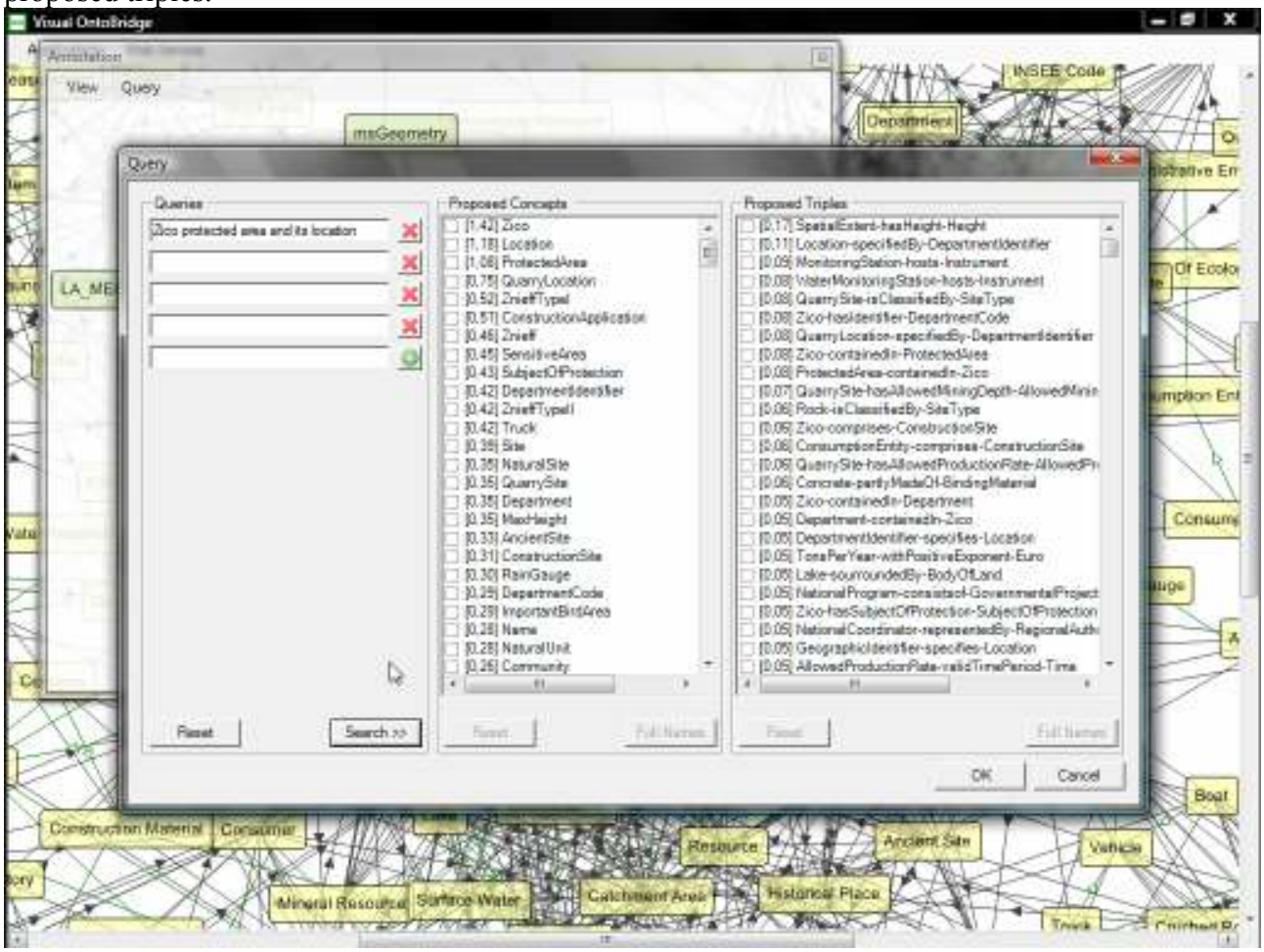


Figure 4-4 Annotating Feature Type

Later the selected entities are inserted into the graphical annotation editor. Then the use completes the annotation by drawing the relations between the domain ontology concepts and schema elements. This is shown in Figure 4-5.

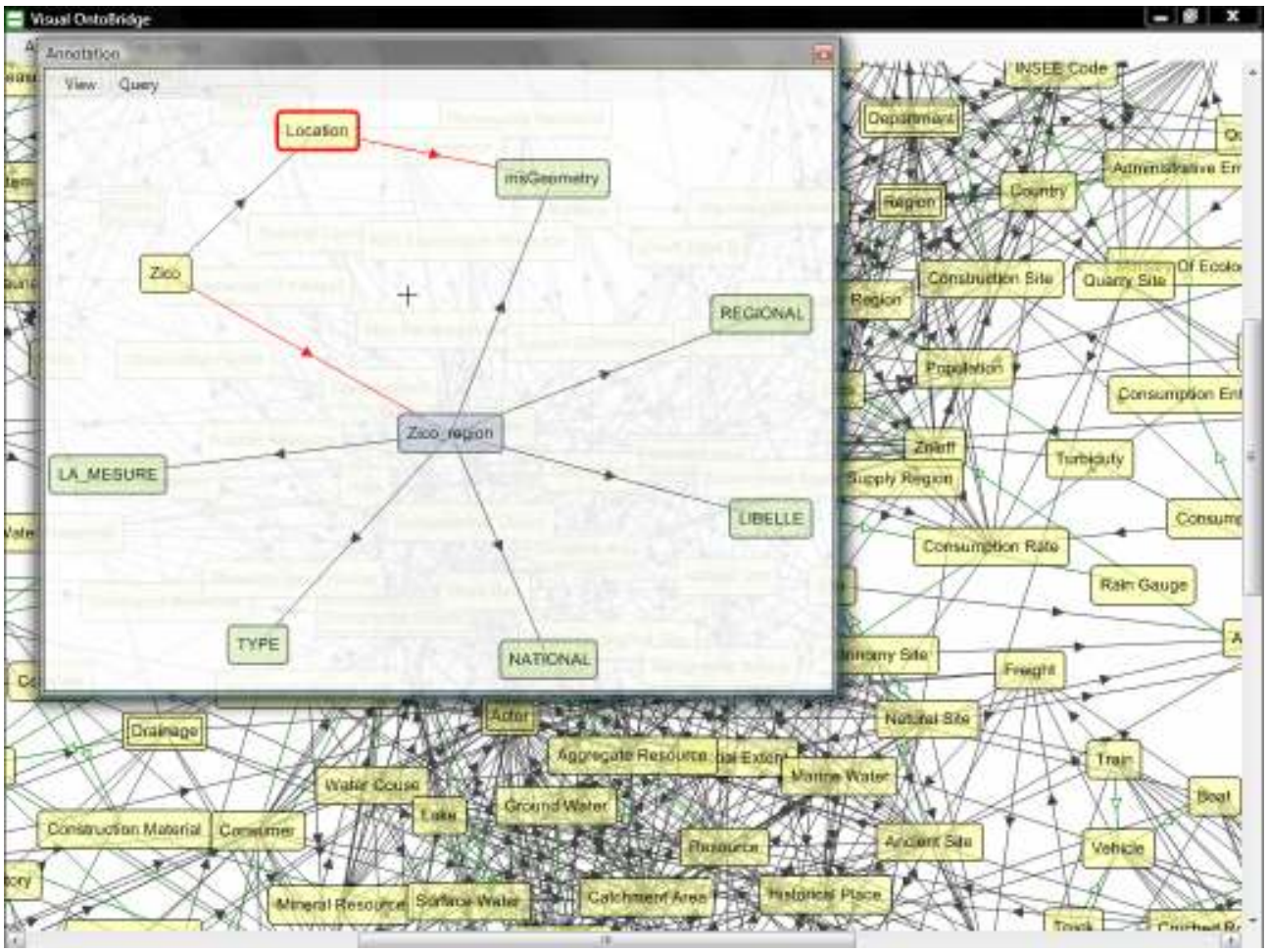


Figure 4-5 Relations between Domain Ontology Concepts and Schema Elements

This chapter presents the SWING system and the need for a natural language interface. Next chapter will discuss some existing approaches for natural language interfaces.

Chapter 5 Analysis of Approaches for Natural Language Interfaces

There exist a few frameworks for semantic Web services discovery, composition and invocation. Chapter 4 discusses the SWING system and the need for a natural language interface. The problem is that the user interface is complex. For instance, the SWING system takes WSML query as input, and WSML is a new programming language which cannot be easily understood. Therefore a simpler interactive solution is demanded. In this chapter, approaches for natural language interfaces are analyzed. Among the different approaches, Question Answering systems (QA) are mainly discussed. Chapter 5.1 introduces QA system, including what it is, the features of QA system, and an instance of a QA system. Chapter 5.2 discusses the use of QA system in original Web.

5.1 Question Answering Systems

In this section, a definition of Question Answering System is given first. The advantages/features of QA systems are then presented. An example of QA systems is given. Then QA systems in different Web scenarios are discussed, which includes evaluations of existing QA systems, the use of QA systems.

5.1.1 What is a Question Answering system?

Wikipedia defines QA systems as follows:

Question answering (QA) [34] is a type of information retrieval. Given a collection of documents (such as the World Wide Web or a local collection) the system should be able to retrieve answers to questions posed in natural language. QA is regarded as requiring more complex natural language processing (NLP) techniques than other types of information retrieval such as document retrieval, and it is sometimes regarded as the next step beyond search engines.

5.1.2 Why a Question Answering system?

Many Web users do not have enough knowledge of using a search engine in an efficient way. With a search engine, users have to break the information they want to search into some keyword segments which requires more or less some skills. With Question answering systems, users can just insert natural language query as input, and QA systems will further handle the question and return the answers in natural language to users. In this case, it reduces the complexity of information retrieval for non-technical users.

5.1.3 An Example of QA system – Start

In this section, a real QA system - START is introduced.

START [35] is one of the world's first Web-based question answering system. It has been developed by Boris Katz and his associates of the InfoLab Group at the MIT Computer Science and

Artificial Intelligence Laboratory. As a question answering system, START aims to supply users with "just the right information," instead of merely providing a list of hits. Currently, the system can answer English questions about places (e.g., cities, countries, lakes, coordinates, weather, maps, demographics, political and economic systems), movies (e.g., titles, actors, directors), people (e.g., birth dates, biographies), dictionary definitions, etc.
Figure 5-1 shows the question interface of Start:

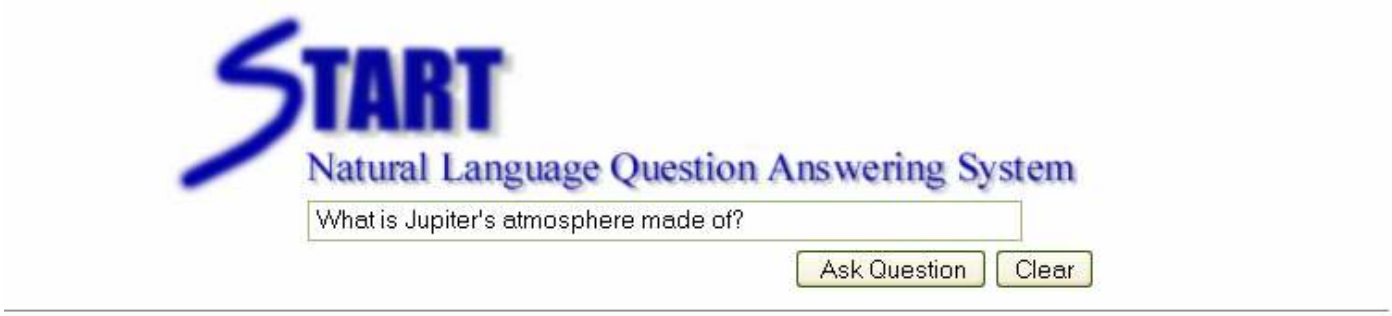


Figure 5-1 Start Question Interface

Figure 5-2 shows the answer interface:

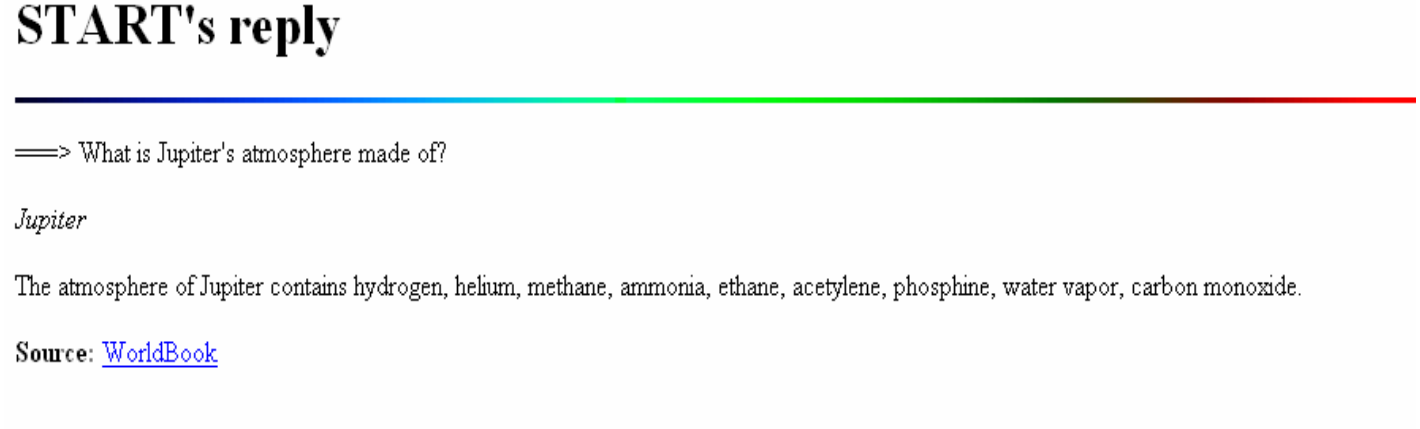


Figure 5-2 Start Answer Interface

5.2 Question Answering Systems in original Web

In this section, QA systems in original Web are discussed. The disadvantages of original Web are presented. Then possibilities of using QA systems in original Web are introduced.

5.2.1 Original Web:

Today's Web is mainly based on (X) HTML, which itself doesn't know the meaning of the resources it marks. The resources on the Web are in many cases scattered in different places randomly. The way of finding the resources are either through keyword-matching based search engine like Google or in-site search on certain portal.

5.2.2 Problems in Original Web

The way of organizing resources in original Web introduced a few disadvantages. Firstly,

computers do not understand the meaning of the web pages. The will lead to difficulties in further processing data resources automatically. Besides, most resources are not structured in a systematic way which makes it hard to find them in an efficient way. The primary way of finding useful info is to use keyword matching search which does sometimes miss useful resources or find useless ones as a result of the limitation of keyword matching.

5.2.3 Question Answering Systems in Original Web:

Some QA systems achieve their process with the help of keyword-matching based search, e.g., Esfinge [36]. Their essential work principle of getting the answer includes two steps:

1) Construct patterns from the question in NL (nature language).
2) Search the patterns from Web using Google Searching engine and analyze the returned searching results to form an answer in NL.

Some QA systems involve semantic annotations in their proceeding process, e.g., InSicht [38].
The mechanism they use is the following:
In query expansion : Equivalent and similar semantic networks are derived from the original query network by means of lexicon-semantic relations from HaGenLex [38]) and a lexical database (GermaNet), equivalence rules, and inferential rules like entailments for situations (applied in backward chaining). The result is a set of disjunctively connected semantic networks that try to cover many possible kinds of representations of sentences possibly containing an explicit or implicit answer to the user's question.

This chapter discussed some existing approaches for natural language interfaces, and mainly talks about QA systems. These QA systems are not chosen to be used as natural language interface for discovery of semantic Web services because QA technologies in semantic Web are not yet mature. Next chapter presents my solution of a simplified natural language discovery of semantic Web services which is based on ontologies. It is the main contribution of the thesis. In the conclusion chapter we will discuss potential future use of QA technologies for this purpose.

Chapter 6 ONALDIS

In chapter 4, the SWING system is introduced and the need for a natural language interface is brought up. In chapter 5, question answering systems are discussed as an existing approach for natural language interface. In this chapter an approach of discovery of semantic Web services is presented. It is called ***Ontology Based Natural Language Discovery of Semantic Web Services*** (ONALDIS). The features of this approach are, firstly it is based on domain ontologies, and secondly, it is specially used for discovery of semantic Web services.

Is it possible to create an ontology based natural language interface for discovery of SWS in the geospatial domain? This question also focuses on the primary hypothesis in this thesis work. According to the sub hypothesis, this can be achieved by accomplishing the following tasks:

- There is a system which takes natural language questions as input and generates WSML queries which can query SWS in geospatial domain in order to simplify the querying process to SWS.
- The output of the above can be the input of some Semantic Web services frameworks which can discovery, compose and invoke SWS in geospatial domain

Chapter 6.1 presents the theory of mapping from query in natural language to WSML query, and gives the concrete implementation of the approach and the integration with SWING system. It is also the main contribution of this thesis work.

Chapter 6.2 discusses the possibility of applying ONALDIS to other ontology domains.

6.1 Analyze Questions – Map Questions in Natural Language to WSML Query

The purpose of question analysis is to translate queries in natural language which are understandable for human-being to formal queries which can be further processed by computer. This is a process which achieves translation from “informal” to “formal”.

In order to achieve the above purpose, the first issue is to decide what kind of formal language can be used to describe NL question. Since the query is for querying semantic Web services, the alternatives could be for instance WSML, OWL-S, and DAML-S. In this thesis work WSML is chosen to implement the formal queries, because it is currently one of the most popular and wild used Web service modeling languages in SWS research area. In addition, the domain ontologies that are used in my ONALDIS are written in WSML as well.

6.1.1 General description - Goal and main idea

In order to map questions to WSML query, the first issue is about to understand the question. There are a lot of relevant researches in NL processing area which take care of this issue. Inspirited by these approaches, the main idea is brought up to solve the problem in the context of this thesis.

The idea is to make some example questions and their corresponding WSML queries, from where some “hidden patterns” can be found in NL questions which can help with the mapping (from

question to WSML query). The following are the steps:

a) write some WSML queries according to the NL question, here is an example:
   Question:
   What is the production capacity of quarry X ?
   WSML query:
   ?x[?att hasValue ?y] and
   annot#annotate(?y, ?y1) and
   ?y1 memberOf domain#ProductionCapacity
   annot#annotate(?x, ?x1) and
   ?x1 memberOf domain#Quarry
   ?x1 [hasName hasValue X]

b) By analyzing a set of questions and their corresponding WSML queries, it is found that the keywords in questions are the most important resource to help to understand what the questions are asking about. Even single keyword doesn't make too much sense, but a set of words usually expresses clear logic. For instance, a word set {what, temperature, Oslo, today} obviously expresses information like "What's the temperature in Oslo today", a word set {Where, University, Oslo} expresses info "Where is University of Oslo".

c) Based on the understanding of keywords set (patterns), mapping from NL questions to WSML queries can be achieved by generating WSML queries which can be understood by computer, and generate "word sets" which correspond to WSML queries. By mapping the question to the pre-defined word sets, the questions in natural language can be mapped to WSML queries.

The focus is therefore about how to generate the word sets and the corresponding WSML queries.

6.1.2 Pre-defined WSML queries and keyword sets?

In some of others' previous research work, WSML is used to pre-define formal queries in some systems/platforms. For instance, in paper "Product Catalogue Management with Semantic Web Services" [45], a set of WSML queries associated together with some questions are pre-defined. An example is: a simple question might be 'does my bundle contain a network connection product?' The WSML query for this is 'MyBundle Contains Network Connection?'. Another example is: A simple question might be 'does my bundle contain a network connection product?' The WSML query for this is 'MyBundle Contains NetworkConnection?'.

This pre-defined approached is not adopted in ONALDIS because it has been detected with the following limitations:

- Firstly, for each different ontology domain, a new set of WSML queries have to be defined again. That is to say, the defining process is not reusable. The entire job must be done once and once again manually after changing problem domain.
- Secondly, since the WSML queries are manually defined, it is hard to control the correctness.

In this thesis, WSML queries are generated from the domain ontologies instead of pre-defined. The advantages are:

- It is a higher level "model" which describes the problem domain.
- Both keyword sets and WSML queries can be generated from this model.

Next section discusses how to generate WSML queries and keyword sets from ontologies.

6.1.3 Generate WSML queries and keyword sets from ontologies

This section explains the idea of generating WSML queries and keyword sets from ontologies.

In order to generate WSML queries and keyword sets from ontologies, first ontologies need to be analyzed. In this thesis, the ontologies used are SWING ontology and some relevant universal ontologies. In order to illustrate the relations between ontologies and the keyword sets/WSML queries, diagrams are used for showing more info visually. The following are some examples:

Example 1: Figure 6-1

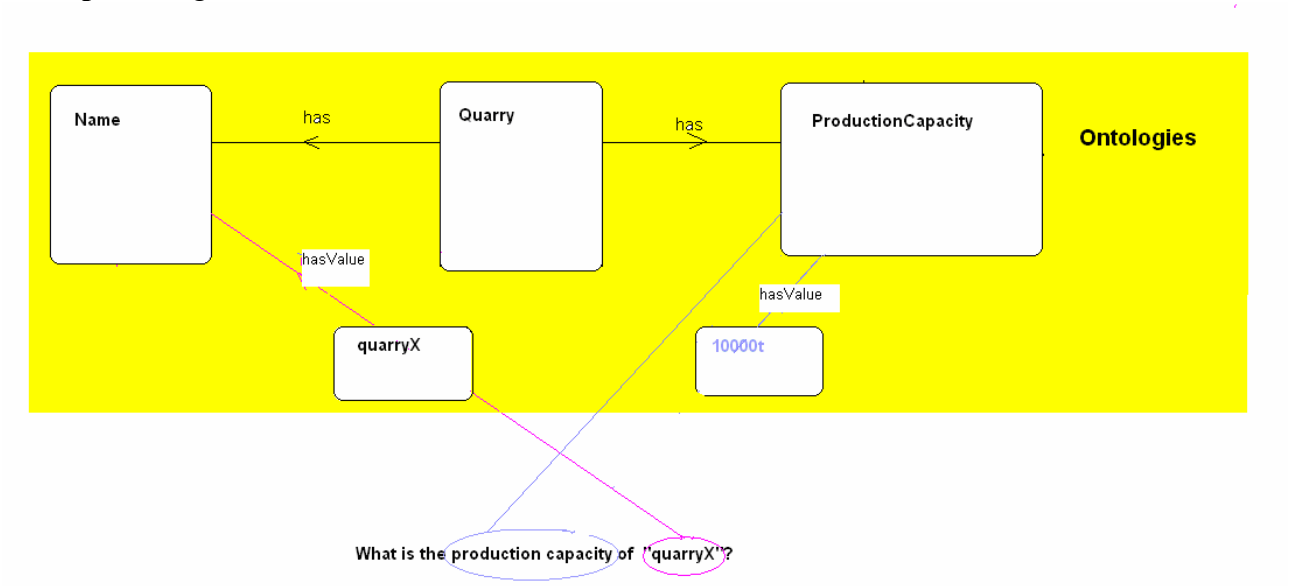


Figure 6-1 Mapping between Keyword Sets and Ontologies, Example 1

From Figure 6-1 it can be observed that this piece of ontology (concepts Quarry, Name, ProductionCapacity and the relations between them) can be generated to keyword set [what, production capacity, quarry name]. It can also be generated to the following WSML query which corresponds to the keyword set:

?x[?att hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ProductionCapacity
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry
?x1 [hasName hasValue X]

Example 2: Figure 6-2

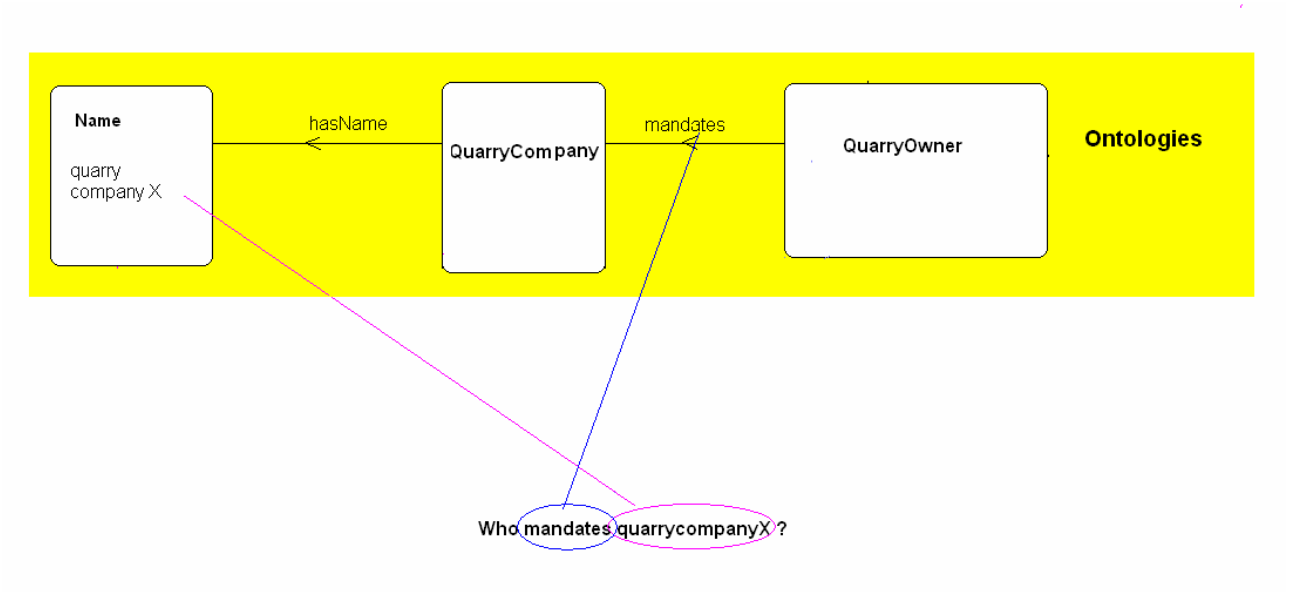


Figure 6-2 Mapping between Keyword Sets and Ontologies, Example 2

From Figure 6-2 it can be observed that this piece of ontology can be generated to keyword set [who, mandates, quarry company name]. It can also be generated to the following WSML query which corresponds to the keyword set:

?x[mandates hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#QuarryCompany
?y1 [hasName hasValue Y]
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryOwner

Example 3: Figure 6-3

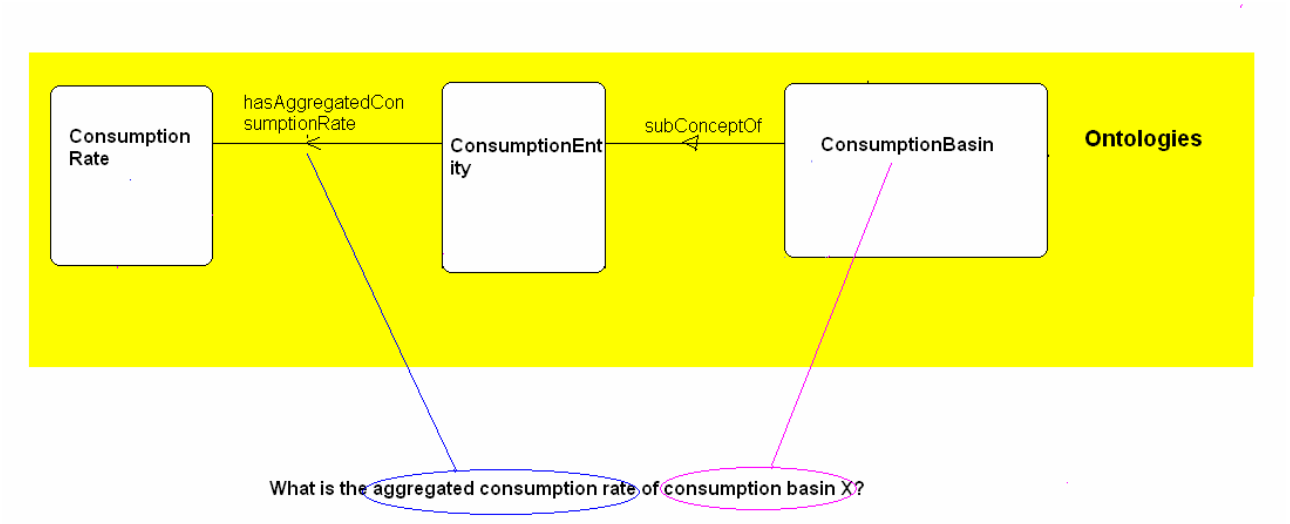


Figure 6-3 Mapping between Keyword Sets and Ontologies, Example 3

From Figure 6-1 it can be observed that this piece of ontology can be generated to keyword set [what, aggregated consumption rate, consumption basin name]. It can also be generated to the following WSML query which corresponds to the keyword set:

?x[?att hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain# AggregatedConsumptionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain# ConsumptionBasin
?x1 [hasName hasValue X]

According to the three examples shown above, it shows that there are relationship between the ontology and keyword sets and the relationship between the ontology and WSML queries. The next section presents the generic rules which take use of these relationships in order to define the way of mapping from the ontology to keyword sets and WSML queries.

6.1.4 Rules for generating WSML query and word sets from ontologies

The following are the rules which are used to generate WSML query and word sets from ontologies.

Rule 1:

This rule defines a "has" relation. "A" and "B" are two concepts in domain ontology. "hasXxYyZz" is the relation between concepts "A" and "B". It can be mapped to the following keyword set and WSML query:

A ------hasXxYyZz.. ---→ B (instanceOf B)
=>
What .* xx yy zz .. of A X?

?x[?att hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain# B
annot#annotate(?x, ?x1) and
?x1 memberOf domain# A
?x1 [hasName hasValue X]

In the keyword set, "X" is a name.
The WSML query explains that "?x" is an attribute which has value "?y". "?y" can be annotated with "?y1". "?y1" is a member of domain "B". "?x" can be annotated with "?x1". "?x1" is a member of domain "A", "?x1" has name "X".

Example 1 of rule 1:

Quarry------hasProductionCapacity---→ProductionCapacity
=>
What .*production capacity of quarry X

?x[hasProductionCapacity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain# ProductionCapacity
annot#annotate(?x, ?x1) and
?x1 memberOf domain# Quarry

?x1 [hasName hasValue X]

Example 2 of rule 1:

QuarryProduct------hasPrice---→AggregatePrice
=>
What .*price of quarry product X

?x[hasPrice hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain# AggregatePrice
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryProduct
?x1[hasName hasValue X]

Rule 2:

This rule defines a "verb" relation. (a verb can be "owns", "locates", "mandates" and so on.) "A" and "B" are two concepts in domain ontology. "xxYyZz" is the relation between concepts "A" and "B". It can be mapped to the following keyword set and WSML query:

A ------xxYyZz.. ---→ B
=>
.* xx yy zz B?

?x[xxYyZz hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#B
?y1[hasName hasValue Y]
annot#annotate(?x, ?x1) and
?x1 memberOf domain#A

Example of rule 2:

QuarryOwner------mandates--→QuarryCompany
=>
.*mandates quarry company Y

?x[mandates hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#QuarryCompany
?y1 [hasName hasValue Y]
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryOwner

Rule 3:

This rule defines a “subset” relation. “A’” is a sub concept of “A”, therefore the relation between “A” and “B” can also be applied on “A’” and “B”.

A’ ------subConceptOf------ A------xx.. ---→ B
=>
A’ ------xx.. ---→ B

Example:

ConsumptionBasin----subConceptOf----
ConsumptionEntity----hasAggregatedConsumptionRate --→ConsumptionRate
=>
ConsumptionBasin----hasAggregatedConsumptionRate --→ConsumptionRate

Obviously the stricter the rules are defined, the more accuracy the out coming WSML queries and word sets will be. Therefore the rule defining is very pivotal in the whole WSML queries/keyword sets generating process. This could be a possible further work of this thesis.

6.1.5 Difficulties in question analyzing

The difficulties in question analyzing is mainly from two aspects:

Firstly, in order that computer could understand the question, a number of word sets and corresponding explanation sets need to be generated from ontologies. The questions could be any question in the real world; and each question can be expressed in several ways. Defining rules for matching as much questions as possible is challenging.

Secondly, since WSML queries/ word sets are generated from ontologies, the definition of ontologies can directly affect the quality of the WSML queries/ word sets generating, and in many situations, the ontologies are defined by some other organization or group of people. The quality can not always be guaranteed. A standard is required.

6.1.6 The domain of questions

Relatively, analyzing questions in a specific domain would be easier, firstly because the number of questions is much reduced in a specific domain comparing to those without domain. Therefore the number of the possible keyword sets should be much smaller. Secondly, the ontologies in the specific domain will be much simpler than in a wider domain. Therefore the rules for generating WSML queries/ keyword sets will be relatively simpler.

6.1.7 Implementation in the SWING domain

In this section it is illustrated how to generate keyword sets and WSML queries from ontologies by using ontologies in the SWING project as an example.

6.1.7.1 Process

The following process is used to address the problem.

1. Find possible questions in SWING.
2. Find possible word sets according to the ontologies in SWING domain and questions, which will help to define the rules to generate keyword sets from ontologies.
3. Find the corresponding WSML queries according to the ontologies in SWING domain and questions, which will help to define the rules to generate WSML queries from ontologies.
4. Define rules (patterns) which can be used to generate WSML queries and word sets from ontologies.
5. Generate word sets from ontologies.
6. Generate corresponding WSML queries.
7. Do the mapping between question in NL and WSML query according to 5 and 6.

Each of the steps above will be discussed:

1. Find possible questions in SWING

The following are a few questions which might be solved in SWING ontology domain.
1. What is the location of quarries X?
2. What is the address of industrial site X?
3. What is the population of regional X?
4. Who mandates construction company Y?
5. What is the industrial activity of quarry X?
6. What is the population of country Y?
7. What is the production rate of exploitation Z?
8. What is the price of quarry product Y?
9. What is the functional property of quarry product Y?
10. What is determined by rock type X?
11. What determines allowed production rate?
12. What activity involves activity sawing?
13. Who administrates quarry x?

2. Find possible word sets according to the ontologies in SWING domain and questions

The following are the keyword sets according to the NL questions in 6.1.7.5. The principle here is that the questions should be distinguished from others by applying keyword sets on them.
1) What .*location of quarry .*
2) What .*address of industrial site .*
3) What .*population of regional .*
4) .*mandates construction company .*
5) What .*industrial activity of quarry .*
6) What .*population of country .*
7) What .*production rate of exploitation .*
8) What .*price of quarry product .*
9) What .*functional property of quarry product .*
10) .*determined by rock type .*
11) .*determines allowed production rate .*
12) .*involves activity sawing .*
13) .*administrates quarry .*

3. Find the corresponding WSML queries according to the ontologies in SWING domain

and questions

Here a few examples of WSML queries which correspond to questions above are defined as following:

1. Question: What is the location of quarry X ?

   WSML Query:

   ?x[hasLocation hasValue ?y] and
   annot#annotate(?y, ?y1) and
   ?y1 memberOf domain#Location
   annot#annotate(?x, ?x1) and
   ?x1 memberOf domain#Quarry
   ?x [hasName hasValue X]

2. Question: What is the adress of industrial site X ?

   WSML Query:

   ?x[hasAdress hasValue ?y] and
   annot#annotate(?y, ?y1) and
   ?y1 memberOf domain#Address
   annot#annotate(?x, ?x1) and
   ?x1 memberOf domain#IndustrialSite
   ?x [hasName hasValue X]

More examples can be found in appendix A.

4. Define rules

The rules have been explained in previous section:

Rule 1:

A ------hasXxYyZz.. ---→ B (want instanceOf B)
=>
What .* xx yy zz .. of A X?

?x [xxYyZz hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain# B
annot#annotate(?x, ?x1) and
?x1 memberOf domain# A
?x1 [hasName hasValue X]

Rule 2:

A ------xxYyZz.. ---→ B
=>
.* xx yy zz B?

?x[xxYyZz hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#B
?y1[hasName hasValue Y]
annot#annotate(?x, ?x1) and
?x1 memberOf domain#A

Rule 3:

A' ------subConceptOf------ A------xx.. ---→ B
=>
A' ------xx.. ---→ B

5. Generate word sets from ontologies.

Use the program to generate keyword sets from ontologies.
See appendix D.

6. Generate corresponding WSML queries.

Use the program to generate WSML queries from ontologies.
See appendix D.

7. Do the mapping between question in NL and WSML query according to 5 and 6

The mapping between question in NL and WSML query is shown in next section.

6.1.7.2 Mapping program (implemented in Java)

Figure 6-4 shows the work flow of ONALDIS program.

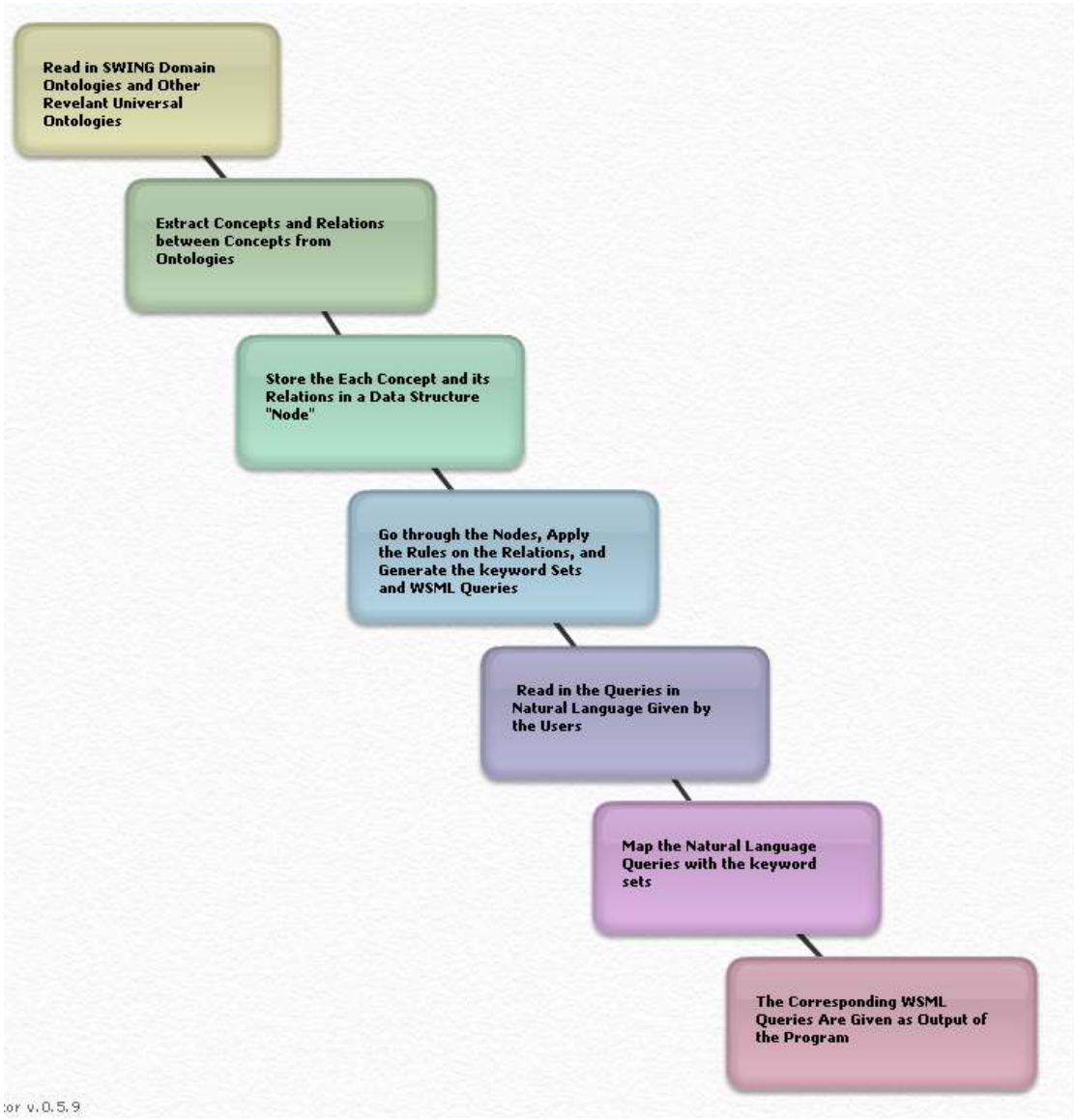


Figure 6-4 ONALDIS Work Flow

Figure 6-5 shows the class diagram of ONALDIS, with the important variables and methods in each class. The program includes three classes – ReadingOntology, MappingApplication, and Node. To use it for mapping, class ReadingOntology should be run first, and then class MappingApplication can run.

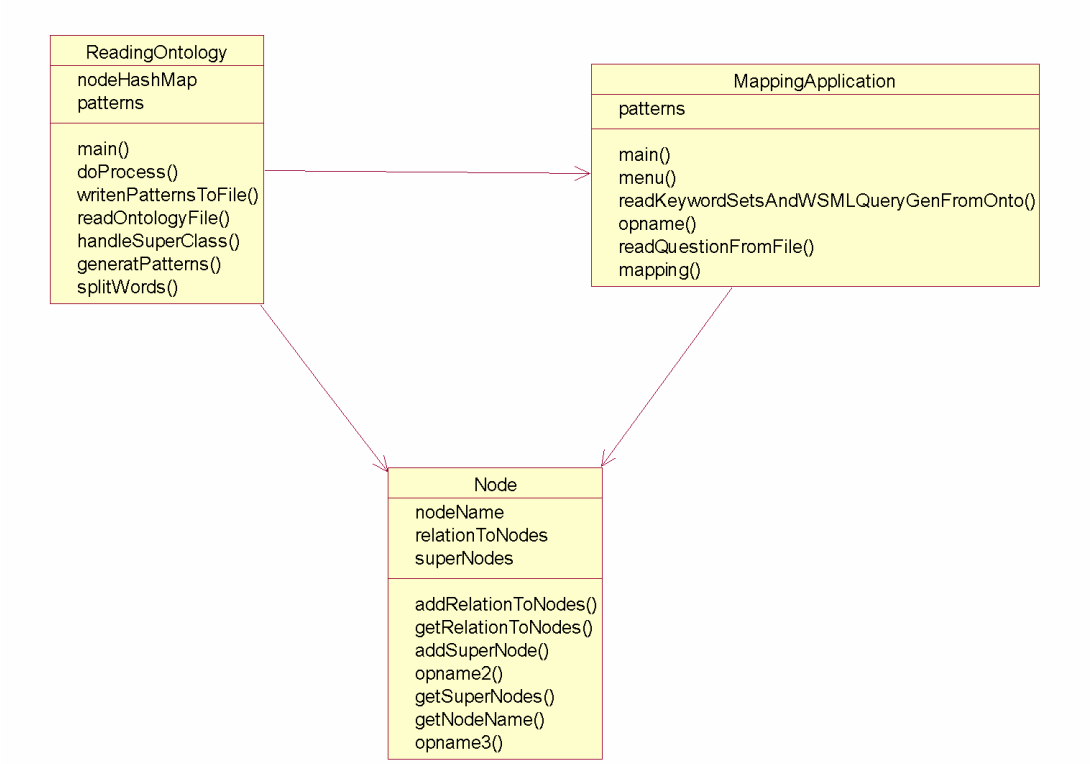


Figure 6-5 ONALDIS Class Diagram

Class ReadingOntology mainly includes the following methods:

1. doProcess: It is a method which includes all the process of reading ontologies.
2. readOntologyFile: It reads ontologies written in WSML from a file line by line. While reading in the ontology, it extract the keyword and notations in WSML, for instance, “concept”, “subConceptOf”, “impliesType”, “//”, “#”, “{}” and so on. What this part deos is similar to what code compilers do. It tries to understand the WSML code as the way a compiler does, and picks up the interesting information, then generate
   The concepts are presented as objects of class Node. An important attribute in reading ontology is “nodeHashMap”. It is a hash map which stores all the ontology concepts. While reading ontology, it is checked if a concept already exists in the node hash map. If not, the new concept will be stored in the node hash map. While reading the ontology, if the keyword “concept” is detected, it will be stored in the node hash map. If the keyword “subConceptOf” is detected, the concept’s super concept will be stored in the node attribute. All of the super nodes are stored in a array list called “supernodes”. If the keyword “impliesType” is detected under a concept defination, it is known that the phrase before the keyword “impliesType” is a name of relation, and the phrase after the keyword “impliesType” is the related concept of the current concept. All the related concepts of the current concept will be stored in a hash map inside a node called “relationToNodes”.

3. writePatternsToFile: this method writes all patterns generated from ontology into a file.
4. handleSuperClass: this method copy all the relations a super node has to its sub node.
5. generatePatterns: this method goes through the node hash map, and for each relation, the rules in section 6.1.4 are applied. The construction of a relation decides which rule should be applied. If the relation is started with “has”, rule 1 will be applied, or rule 2 will be applied. The keyword sets and WSML queries are generated by extracting the info from the node name, relations of the node, and the related node.

Class MappingApplication mainly includes the following methods:

1. menu: This method provides a command line based user interface.
2. readKeywordSetsAndWSMLQueryGenFromOnto: This method reads in the keyword sets and corresponding WSML queries generated from ontologies in ReadingOntology class.
3. readQuestionFromFile: This method provides user the possibility of reading in a list of questions from a file.
4. mapping: This method generates the result which can be shown on screen.

The program is command line based. It takes the names of WSML ontologies files as run arguments. During the run of program, it takes question(s) as input, and gives WSML query/queries as output. User can choose to enter one question in command line or a name of file which includes one or more questions as input. And also user can choose to show the output WSML query/queries in command window or write the result(s) into file.

The following shows an example of running the program:

```
*******Options*******************************
    1) Read questions from file
    2) Read questions from keyboard
    3) Exit
Enter your option:1
Enter question file name:(question.txt)
question.txt
*******Options*******************************
    1) Write answer into file
    2) Show answer on screen
    3) Exit
Enter your option:2
Question: What is the location of quarry X

WSML Query:

?x[hasLocation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Location
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry
?x [hasName hasValue X]
```

More run examples can be found in appendix C.

Here is a description of how the program works, based on the different parts of the program.
The program is written in Java language. It contains two parts:

1. One part is called “ReadingOntology”. It reads in the domain ontologies and other revelant universal ontologies. The relations between the ontology concepts are analyzed. Then keyword sets and WSML queries are generated from ontologies.

2. The other part is called “MappingApplication”. What it does is to read in the queries in natural language, and then maps it to keyword sets and WSML queries which are generated from part 1.

The program can be found in Annex D.

Flexibility

This program can be applied to any mapping from questions in natural language to WSML queries, as long as suitable ontology files are provided. A suitable ontology file means it must fulfill two

conditions: first, it must be written in WSML; second, the WSML language grammar must be the same as the ones that is used in the SWING ontology. If the two conditions are not fulfilled, the program has to be adjusted before being used.

Similar system/programs
There exists other similar mapping program. For instance, "ChartParser" [29] is a program which can map the natural language sentences to a grammar tree. The reason that it is not used in this research work for mapping is that "ChartParser" is meant to be used to map natural language/or any other kinds of queries to a tree format according to user defined rule, while my purpose is to analyze ontology described in WSML and further map them to keyword patterns and WSML queries.

Response for out-of-domain questions
In this implementation, the boundary of questions is defined inside Quarry ontology ("QuarriesOntology.wsml") domain. For any out-of-domain questions, the program is able to handle them in a user friendly way. For example, the program gives the following response on screen:

```
Sorry, the program can not answer your question; the reason might be the
following:
    1) The words in question are wrongly spelled.
    2) The question is out of the specified domain.

Please try other questions.
```

6.1.7.3 What can ONALDIS do in the SWING domain and other domains

Since the rules are embeded in the ONALDIS program code, the whole mapping process from queries in natural language to WSML query for discovery semantic Web services can be achieved automatically. User has to only provide two things, one is the domain ontologies and other relevant universal ontologies which are described in WSML, and the other is the queries in natural language. The ONALDIS program automatically process the query and gives WSML queries as output.

If ONALDIS is to be applied on other ontologies domains, it requires some manual work which is to re-define the rules which suit the new ontologies in the program code.

6.2 Apply ONALDIS to Other Ontology Domains

ONALDIS can be applied to other ontology domains. The ontology must satisfy the following conditions:

1. The ontology must be defined with WSML.
2. The convention of defining the ontology must be the same as SWING ontology.
   (The convention of defining the ontology is for instance the use of verb in a relation, or the use of capital letter for the first letter in a concept. For example, in relation "hasProductionCapacity", verb "has" is used for an ownership relation, "Production", "Capacity" starts with capital letters.)
3. If the convention of defining the ontology is different from SWING ontology, the rules must be adapted in order to apply ONALDIS to the new ontology.

This chapter presents ontology based natural language discovery of semantic Web services. Next chapter will illustrate how ONALDIS is applied on SWING system.

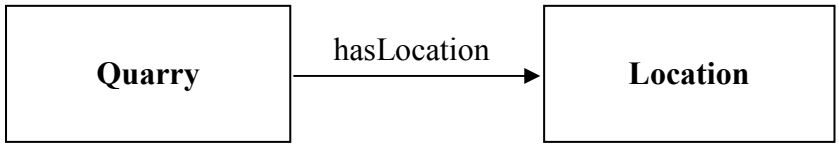
Chapter 7 ONALDIS Applied on SWING

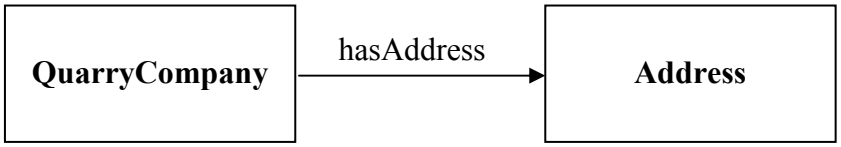
In this chapter, it is discussed how to apply ONALDIS on the SWING system. First adapting ONALDIS to fit the service searching and registration in SWING project is introduced. Then the other possibilities with a natural language interface in SWING system are discussed, and evaluations based on the hypotheses are presented.

7.1 Adapting ONALDIS to fit the service searching and registration in SWING

The idea here is to replace the keyword search with a natural language query, in addition, to wrap the three separated parts - WSML goal, keyword and the bounding box into one simple query box where user can type in the question they want to ask in natural language.

As illustrated in Chapter 6, ONALDIS aims to read in and analyze a group of ontologies, and further generate keyword sets and relevant WSML queries from ontologies. In order to adapt ONALDIS to meet the needs of SWING's interface, a set of rules which will be used to generate keyword sets and WSML queries have to be defined according to the problem domain that SWING wants to solve.
In order to automatically retrieve the info that is defined by the bounding box, those properties that are related to geospatial terms should be taken care of. For instance:





These geospatial related properties will be generated to constraints according to the rules and added in WSML query.

Here is an example:

Figure 7-1 shows the ontology:

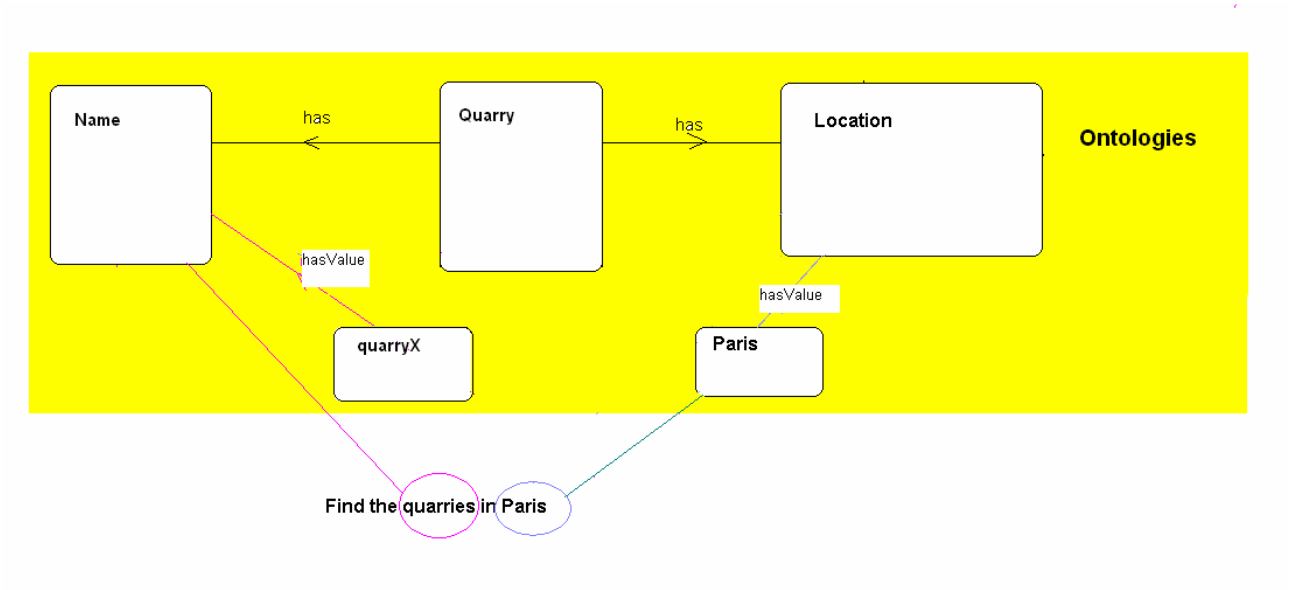


Figure 7-1 Ontology

Keyword set regarding to this
Find * quarries in *

WSML query:

?x[hasName hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Name
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry
?x [hasLocation hasValue X]

The rule behind could be the following:

Name<----- hasName -----A ------hasLocation ---→ B (want instanceOf B)
=>
Find/give .* A in/at *

?x[hasLocation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain# Location
annot#annotate(?x, ?x1) and
?x1 memberOf domain# A
?x1 [hasName hasValue X]

7.2 Other Possibilities with NL Interface

The natural language interface can also be speech based. There exists some software in market which can convert speech to text. The converted text can further be mapped to WSML query.

7.3 Evaluation

In this section, the results regarding to the hypothesis formulated in chapter 2 are presented. The advantages and limitations of the ONALDIS solution are then discussed.

7.3.1 Evaluation results

The evaluation is based on the hypothesis table introduced in chapter 2.1. The results are shown in Table 7-1.

	Hypothesis	Result
H	It is possible to create an ontology based natural language interface for discovery of SWS in the geospatial domain.	Theoretical True/Not proven
H 1.1	It is possible to create a system with the help of ontologies which takes NL questions as input and generate WSML queries which can query SWS in geospatial domain in order to simplify the querying process to SWS.	Theoretical True/Not proven
H 1.1.1	It is possible to create a system with the help of ontologies which takes NL questions as input and generate WSML queries which can query SWS in SWING domain.	True
H 1.1.2	H1.1.1 can be applied to different ontologies domains.	Theoretical True/Not proven
H 1.1.3	The accuracy of the answer through the natural language interface is as good as using a specialized query language.	Not proven
H 1.1.4	Natural language interface is easier to use for end users.	True
H 1.2	The output of H1.1 can be the input of some Semantic Web services frameworks which can discover SWS in geospatial domain.	True

Table 7-1 Evaluation Based on Hypothesis

The innovation part of this thesis is to prove the hypothesis H1.1.1, which is the possibility of creating a system with the help of ontologies which takes questions in natural language as input and generate WSML queries which can query semantic Web services in SWING domain. Other hypothesis which are proven true are H1.1.4, which is natural language interface is easier to use for end users, and H1.2, the output of H1.1 can be the input of some Semantic Web services frameworks which can discovery SWS in geospatial domain.
Some hypotheses are not proven with real examples, but explained in theory. Among these, hypothesis H1.1.2 which is H1.1.1 can be applied to different ontologies domains is discussed in chapter 6.3.
Hypothesis H1.1.3 which is the accuracy of the answer through the natural language interface is as good as using a specialized query language is not proven.

The accuracy of the answer through the natural language interface can not be as good as using WSML query directly because of the mapping process between natural language query and WSML query. The following reasons influence the accuracy of mapping:

1. Definition of ontologies.
   The important step is to generate the word sets and WSML queries from domain ontologies and other general ontologies. ONALDIS requires the ontologies must be defined with WSML, and the convention of defining the ontology must be restrict, which is to say no grammar error and so son.
2. Definition of the mapping rules
   Rules definition can also influence the accuracy of mapping. The rules need to be adjusted according to the definition of ontologies.

7.3.2 Advantage and Limitations

The advantage of this approach is that it can be adjusted for a wider domain. Geospatial domain is not the only domain the approach will work for.
One of the limitations is that it is limited to the mapping of questions to one prefabricated keyword set. As the ontologies grow larger the number of possible keyword sets will grow exponentially, and it will be close to impossible to match all possible questions that can be asked against a pre-computed set. This is thus not a feasible long term solution. A more general QA technology could be a better solution to improve this limitation. This will be further discussed in chapter 8.3.
The other limitation is about the accuracy of mapping which is discussed in previous section. In order to improve the limitations, more elaborated mapping rules need to be refined.

This chapter illustrates applying ONALDIS on the SWING system, with evaluation of ONALDIS based on the hypotheses.

Chapter 8 Conclusion and Further Work

This chapter concludes the thesis work and discusses the future extensions with QA system.

8.1 Conclusion

The thesis explores the possibility of providing the user with a convenient interface based on a semantic Web services processing framework, and to process natural language questions which helps to discover semantic Web services in the geospatial domain. With the help of ontologies, the interface will take queries in natural language as input and generate WSML queries as output, which can query SWS in geospatial domain in order to simplify the querying process for semantic web services. The SWING system is used as a semantic Web services processing framework in this thesis.

8.2 Contribution

The main contribution of this thesis is to design and implement an Ontology Based Natural Language Discovery of Semantic Web Services (ONALDIS) based on the SWING framework. ONALDIS works generally in the geospatial domain, and with the implementation in chapter 7, it has been proven that it works in SWING ontology domain. It offers a command line interface, takes natural language questions as input, and gives WSML queries as output. This WSML query is a formal query to discover semantic Web services.

The internal/background mechanism is that it first read in the domain ontologies and other relevant ontologies. By applying the rules, it analyzes these ontologies and generates two groups of information. One is word sets and the other is WSML queries. When the users enter a question in natural language, the question is mapped to one of the keyword set, and the corresponded WSML query is further found.

ONALDIS has proven the earlier hypotheses which are:

1. It is possible to create a system with the help of ontologies which takes NL questions as input and generate WSML queries which can query SWS in SWING domain.
2. Natural language interface is easier to use for end users.
3. WSML queries generated from natural language questions can be the input of some Semantic Web services frameworks which can discovery SWS in geospatial domain.

A benefit with ONALDIS's ontology analyzing process is that it helps to map to synonyms which makes it possible to find matches for different words that are synonyms for each other. For instance, different rock types, such as granite, can be recognized.

8.3 Limitations

ONALDIS shows that the concepts from an ontology are very useful for analysing natural language queries. Meanwhile the technical approach of ONALDIS still has two obvious limitations. The first one is that it is limited to the mapping of questions to one prefabricated keyword set. As the

ontologies grow larger the number of possible keyword sets will grow exponentially, and it will be close to impossible to match all possible questions that can be asked against a pre-computed set. A more general QA technology could be a better solution to improve this limitation. The other limitation is about the accuracy of mapping. In order to improve the limitations, more elaborated mapping rules need to be refined.

Figure 8-1 shows an example of a piece of course ontology. It can actually answer the question "Who in school X can Java programming?" In order to generate keyword sets and WSML queries from this course ontology regarding to this question, more complex rules are supposed to be defined. This can be one of the further work.

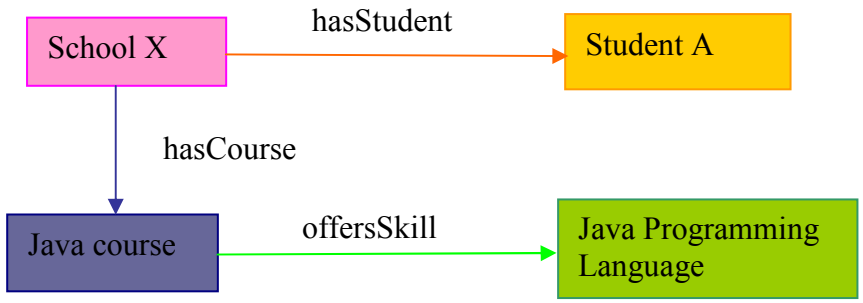


Figure 8-1 A Piece of Course Ontology

8.4 Future Extensions with QA System

Chapter 5 introduces question answering systems as approaches for natural language interfaces. With the development of semantic Web, some semantic elements are introduced into QA systems. Due to the limitations of ONALDIS solution, a more sophisticated query answer system for a more generalized solution would be a goal for the future. Ontology based QA systems seems to be a promising basis for future solutions.

8.4.1 Question Answering Systems in Semantic Web

There exists some Ontology-driven QA system. AquaLog[30] is one of them. The main processes of AquaLog include:

1) Translate the question in NL into a set of triples of the form<subject, predicate, object>
2) Construct ontology-compliant queries from the triples.
3) Map the queries with Ontologies/ Knowledge Bases to get the answer.

Until now, there exist few articles or research papers which discuss the possibilities of using QA

system as natural language interface for semantic Web Services discovery, composition, and invocation. The use of QA systems for this would be a relevant strategy for future evolution in this area.

8.4.2 Using Question Answering System for Processing Semantic Web Services

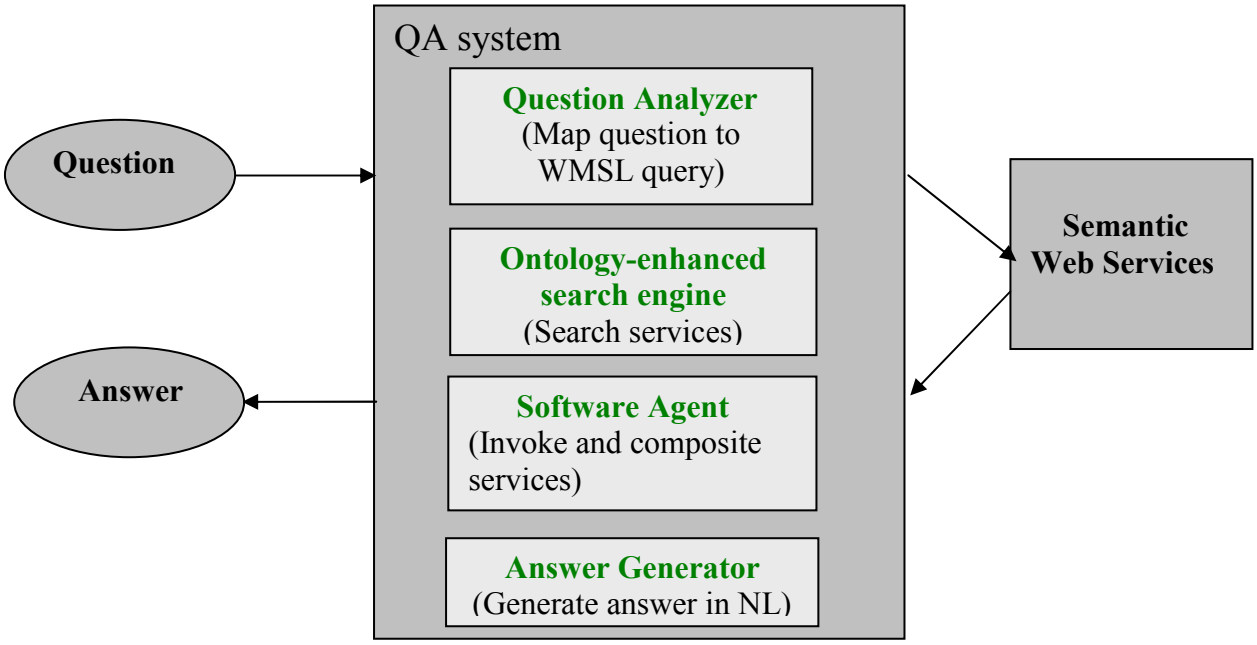


Figure 8-2 Combine QA System and Semantic Web Services

Figure 8-2 illustrates the possibility of adapting question answering system for semantic Web service processing. Ontology-based Question Answering system can be used as a Question Analyzer, an Ontology-enhanced Semantic Web Services search engine, a software agent, and an Answer generator.

As a strategy for further evolution from the results of this thesis, it is recommended to investigate the further use of QA systems, as a basis for a more generalized approach.

Appendix

Appendix A – Natural Language and WSML Querie Pairs

Appendix A shows some natural language queries and then the corresponding WSML queries that is being generated. It refers to section 6.1.7.1.

1. Question: What is the location of quarry X ?

   WSML Query:

   ```
   ?x[hasLocation hasValue ?y] and
   annot#annotate(?y, ?y1) and
   ?y1 memberOf domain#Location
   annot#annotate(?x, ?x1) and
   ?x1 memberOf domain#Quarry
   ?x [hasName hasValue X]
   ```

2. Question: What is the adress of industrial site X ?

   WSML Query:

   ```
   ?x[hasAdress hasValue ?y] and
   annot#annotate(?y, ?y1) and
   ?y1 memberOf domain#Address
   annot#annotate(?x, ?x1) and
   ?x1 memberOf domain#IndustrialSite
   ?x [hasName hasValue X]
   ```

3. Question: What is the population of regional X ?

   WSML Query:

   ```
   ?x[hasPopulation hasValue ?y] and
   annot#annotate(?y, ?y1) and
   ?y1 memberOf domain#Population
   annot#annotate(?x, ?x1) and
   ?x1 memberOf domain#Regional
   ?x [hasName hasValue X]
   ```

4. Question: Who mandates construction company Y ?

WSML Query:

```
?x[mandates hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConstructionCompany
annot#annotate(?x, ?x1) and
?x1 memberOf domain#ConstructionSiteOwner
?y [hasName hasValue Y]
```

5. Question: What is the industrial activity of quarry X ?

WSML Query:

```
?x[hasIndustrialActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Exploitation
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry
?x [hasName hasValue X]
```

6. Question: What is the population of country Y ?

WSML Query:

```
?x[hasPopulation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Population
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Country
?x [hasName hasValue Y]
```

7. Question: What is the production rate of exploitation Z ?

WSML Query:

```
?x[hasProductionRate hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ProductionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Exploitation
?x [hasName hasValue Z]
```

8. Question: What is the price of quarry product Y ?

WSML Query:

```
?x[hasPrice hasValue ?y] and
```

annot#annotate(?y, ?y1) and
?y1 memberOf domain#AggregatePrice
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryProduct
?x [hasName hasValue Y]

9.
Question: What is the functional property of quarry product Y ?

WSML Query:

?x[hasFunctionalProperty hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#FunctionalProperty
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryProduct
?x [hasName hasValue Y]

10.
Question: What is determined by rock type X ?

WSML Query:

?x[determinedBy hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#RockType
annot#annotate(?x, ?x1) and
?x1 memberOf domain#MineralResource
?y [hasName hasValue X]

11.
Question: What determines allowed production rate ?

WSML Query:

?x[determines hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#AllowedProductionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Legislation
?y [hasName hasValue rate]

12.
Question: What activity involves activity sawing ?

WSML Query:

?x[involvesActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Sawing
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Extraction

?y [hasName hasValue involves]

13.

Question: Who administrates quarry x ?

WSML Query:

?x[administrates hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Quarry
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryAdministration
?y [hasName hasValue x]

Appendix B – Keyword Sets and WSML Queries Generated from Ontology

Appendix B shows keyword sets and WSML queries generated from ontologies. It refers to section 6.1.7.1

1) .*contained in department .*
2) What .*estimated consumption rate of infrastructure construction .*
3) What .*reference of legislation .*
4) What .*unit of measure of production capacity .*
5) .*partly made of aggregate.*
6) .*mandates quarry company .*
7) .*contained in regional .*
8) .*determined by rock type .*
9) .*comprises construction site .*
10) What .*unit of measure of basic quantity .*
11) .*performs exploitation .*
12) What .*aggregated consumption rate of consumption entity .*
13) What .*estimated consumption rate of building construction .*
14) What .*location of administrative entity .*
15) What .*production rate of exploitation .*
16) .*administrative part of country .*
17) What .*estimated consumption rate of railway construction .*
18) .*conversion parameters conversion parameter set .*
19) What .*adress of quarry .*
20) .*mandates construction company .*
21) .*administrative part of department .*
22) What .*location of quarry .*
23) .*is quantified by derived quantity .*
24) What .*reference entity of count .*
25) .*contrained by geo# geological formation.*
26) .*determines allowed production rate.*
27) .*with positive exponent count.*
28) .*involves activity sawing.*
29) What .*location of industrial site .*
30) .*administrates quarry .*
31) .*involves activity mixing.*
32) .*reference year time .*
33) .*determines allowed mining depth.*
34) What .*population of community .*
35) What .*estimated consumption rate of road construction .*
36) What .*population of regional .*
37) .*participates in measurement process .*
38) .*subject to transportation .*
39) .*organizes quarry management .*
40) What .*population of country .*

41) .*involves activity processing.*
42) .*hosts instrument .*
43) What .*population of department .*
44) What .*location of country .*
45) What .*location of consumption basin .*
46) What .*aggregated consumption rate of administrative entity .*
47) .*involves activity extraction.*
48) .*contained in country .*
49) What .*functional property of quarry product .*
50) .*involves activity drilling.*
51) What .*type of quarry .*
52) .*partly made of binding material.*
53) .*operates according to sensing procedure .*
54) What .*price of quarry product .*
55) .*involves activity sieving.*
56) What .*production capacity of quarry .*
57) What .*allowed production rate of quarry .*
58) What .*part of measurement .*
59) .*involves activity shoveling.*
60) What .*unit of measure of quantity .*
61) What .*industrial activity of construction site .*
62) .*needs construction material construction material .*
63) What .*location of community .*
64) What .*adress of industrial site .*
65) What .*allowed mining depth of quarry .*
66) .*with negative exponent unit .*
67) .*with positive exponent unit.*
68) What .*aggregated consumption rate of consumption basin .*
69) .*dimension basic quantity .*
70) .*involves activity blasting.*
71) .*contrained by gen# topography.*
72) .*administrative part of quarry management .*
73) What .*industrial activity of quarry .*
74) What .*unit of measure of derived quantity .*
75) What .*location of construction site .*
76) .*with positive exponent population .*
77) What .*measured value of quantity .*
78) What .*measured value of derived quantity .*
79) .*involves activity reassembling.*
80) What .*population of administrative entity .*
81) .*administrative part of regional .*
82) What .*location of regional .*
83) .*related to legislation .*
84) What .*number of count .*
85) .*involves activity washing.*
86) .*manages construction site .*
87) .*valid time period time .*
88) .*counted entity entity .*
89) .*organizes construction site management .*
90) What .*location of department .*
91) .*consumes quarry product .*
92) What .*measured value of basic quantity .*

93) What .*estimated consumption rate of construction .*

1)
?x[containedIn hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Department
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Community

2)
?x[hasEstimatedConsumptionRate hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConsumptionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#InfrastructureConstruction

3)
?x[hasReference hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#LegalDocument
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Legislation

4)
?x[hasUnitOfMeasure hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#TonsPerYear
annot#annotate(?x, ?x1) and
?x1 memberOf domain#ProductionCapacity

5)
?x[partlyMadeOf hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Aggregate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Concrete

6)
?x[mandates hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#QuarryCompany
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryOwner

7)
?x[containedIn hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Regional
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Department

8)
?x[determinedBy hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#RockType
annot#annotate(?x, ?x1) and
?x1 memberOf domain#MineralResource

9)
?x[comprises hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConstructionSite
annot#annotate(?x, ?x1) and
?x1 memberOf domain#AdministrativeEntity

10)
?x[hasUnitOfMeasure hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Unit
annot#annotate(?x, ?x1) and
?x1 memberOf domain#BasicQuantity

11)
?x[performs hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Exploitation
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryOwner

12)
?x[hasAggregatedConsumptionRate hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConsumptionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#ConsumptionEntity

13)
?x[hasEstimatedConsumptionRate hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConsumptionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#BuildingConstruction

14)
?x[hasLocation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Location
annot#annotate(?x, ?x1) and
?x1 memberOf domain#AdministrativeEntity

15)
?x[hasProductionRate hasValue ?y] and
annot#annotate(?y, ?y1) and

?y1 memberOf domain#ProductionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Exploitation

16)
?x[administrativePartOf hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Country
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Regional

17)
?x[hasEstimatedConsumptionRate hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConsumptionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#RailwayConstruction

18)
?x[conversionParameters hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConversionParameterSet
annot#annotate(?x, ?x1) and
?x1 memberOf domain#ConventionalUnit

19)
?x[hasAdress hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Address
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry

20)
?x[mandates hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConstructionCompany
annot#annotate(?x, ?x1) and
?x1 memberOf domain#ConstructionSiteOwner

21)
?x[administrativePartOf hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Department
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Community

22)
?x[hasLocation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Location
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry

23)
?x[isQuantifiedBy hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#DerivedQuantity
annot#annotate(?x, ?x1) and
?x1 memberOf domain#PopulationDensity

24)
?x[hasReferenceEntity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Entity
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Count

25)
?x[contrainedBy hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#geo#GeologicalFormation
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryLocation

26)
?x[determines hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#AllowedProductionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Legislation

27)
?x[withPositiveExponent hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Count
annot#annotate(?x, ?x1) and
?x1 memberOf domain#DensityUnit

28)
?x[involvesActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Sawing
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Extraction

29)
?x[hasLocation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Location
annot#annotate(?x, ?x1) and
?x1 memberOf domain#IndustrialSite

30)
?x[administrates hasValue ?y] and

annot#annotate(?y, ?y1) and
?y1 memberOf domain#Quarry
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryAdministration

31)
?x[involvesActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Mixing
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Processing

32)
?x[referenceYear hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Time
annot#annotate(?x, ?x1) and
?x1 memberOf domain#AllowedProductionRate

33)
?x[determines hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#AllowedMiningDepth
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Legislation

34)
?x[hasPopulation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Population
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Community

35)
?x[hasEstimatedConsumptionRate hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConsumptionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#RoadConstruction

36)
?x[hasPopulation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Population
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Regional

37)
?x[participatesIn hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#MeasurementProcess
annot#annotate(?x, ?x1) and

?x1 memberOf domain#Instrument

38)
?x[subjectTo hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Transportation
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryProduct

39)
?x[organizes hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#QuarryManagement
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryOwner

40)
?x[hasPopulation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Population
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Country

41)
?x[involvesActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Processing
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Exploitation

42)
?x[hosts hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Instrument
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Station

43)
?x[hasPopulation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Population
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Department

44)
?x[hasLocation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Location
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Country

45)

?x[hasLocation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Location
annot#annotate(?x, ?x1) and
?x1 memberOf domain#ConsumptionBasin

46)
?x[hasAggregatedConsumptionRate hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConsumptionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#AdministrativeEntity

47)
?x[involvesActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Extraction
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Exploitation

48)
?x[containedIn hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Country
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Regional

49)
?x[hasFunctionalProperty hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#FunctionalProperty
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryProduct

50)
?x[involvesActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Drilling
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Extraction

51)
?x[hasType hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#SiteType
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry

52)
?x[partlyMadeOf hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#BindingMaterial

annot#annotate(?x, ?x1) and
?x1 memberOf domain#Concrete

53)
?x[operatesAccordingTo hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#SensingProcedure
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Instrument

54)
?x[hasPrice hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#AggregatePrice
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryProduct

55)
?x[involvesActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Sieving
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Processing

56)
?x[hasProductionCapacity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ProductionCapacity
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry

57)
?x[hasAllowedProductionRate hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#AllowedProductionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry

58)
?x[hasPart hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#MeasurementProcess
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Measurement

59)
?x[involvesActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Shoveling
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Extraction

60)
?x[hasUnitOfMeasure hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Unit
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quantity

61)
?x[hasIndustrialActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Construction
annot#annotate(?x, ?x1) and
?x1 memberOf domain#ConstructionSite

62)
?x[needsConstructionMaterial hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConstructionMaterial
annot#annotate(?x, ?x1) and
?x1 memberOf domain#RoadConstruction

63)
?x[hasLocation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Location
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Community

64)
?x[hasAdress hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Address
annot#annotate(?x, ?x1) and
?x1 memberOf domain#IndustrialSite

65)
?x[hasAllowedMiningDepth hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#AllowedMiningDepth
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry

66)
?x[withNegativeExponent hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Unit
annot#annotate(?x, ?x1) and
?x1 memberOf domain#DensityUnit

67)
?x[withPositiveExponent hasValue ?y] and
annot#annotate(?y, ?y1) and

?y1 memberOf domain#Unit
annot#annotate(?x, ?x1) and
?x1 memberOf domain#DensityUnit

68)
?x[hasAggregatedConsumptionRate hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConsumptionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#ConsumptionBasin

69)
?x[dimension hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#BasicQuantity
annot#annotate(?x, ?x1) and
?x1 memberOf domain#DerivedQuantity

70)
?x[involvesActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Blasting
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Extraction

71)
?x[contrainedBy hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#gen#Topography
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryLocation

72)
?x[administrativePartOf hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#QuarryManagement
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryAdministration

73)
?x[hasIndustrialActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Exploitation
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry

74)
?x[hasUnitOfMeasure hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Unit
annot#annotate(?x, ?x1) and
?x1 memberOf domain#DerivedQuantity

75)
?x[hasLocation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Location
annot#annotate(?x, ?x1) and
?x1 memberOf domain#ConstructionSite

76)
?x[withPositiveExponent hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Population
annot#annotate(?x, ?x1) and
?x1 memberOf domain#PopulationDensity

77)
?x[hasMeasuredValue hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Value
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quantity

78)
?x[hasMeasuredValue hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Value
annot#annotate(?x, ?x1) and
?x1 memberOf domain#DerivedQuantity

79)
?x[involvesActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Reassembling
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Processing

80)
?x[hasPopulation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Population
annot#annotate(?x, ?x1) and
?x1 memberOf domain#AdministrativeEntity

81)
?x[administrativePartOf hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Regional
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Department

82)
?x[hasLocation hasValue ?y] and

annot#annotate(?y, ?y1) and
?y1 memberOf domain#Location
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Regional

83)
?x[relatedTo hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Legislation
annot#annotate(?x, ?x1) and
?x1 memberOf domain#AdministrativeEntity

84)
?x[hasNumber hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Number
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Count

85)
?x[involvesActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Washing
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Processing

86)
?x[manages hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConstructionSite
annot#annotate(?x, ?x1) and
?x1 memberOf domain#ConstructionSiteManagement

87)
?x[validTimePeriod hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Time
annot#annotate(?x, ?x1) and
?x1 memberOf domain#AllowedProductionRate

88)
?x[countedEntity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Entity //e.g.
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Count

89)
?x[organizes hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConstructionSiteManagement
annot#annotate(?x, ?x1) and

?x1 memberOf domain#ConstructionCompany

90)
?x[hasLocation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Location
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Department

91)
?x[consumes hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#QuarryProduct
annot#annotate(?x, ?x1) and
?x1 memberOf domain#RoadConstruction

92)
?x[hasMeasuredValue hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Value
annot#annotate(?x, ?x1) and
?x1 memberOf domain#BasicQuantity

93)
?x[hasEstimatedConsumptionRate hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ConsumptionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Construction

Appendix C – Run Example of ONALDIS Applied on SWING

Appendix C shows run example of ONALDIS applied on SWING. It refers to section 6.1.7.2

```
*******Options*******************************
    1) Read questions from file
    2) Read questions from keyboard
    3) Exit
Enter your option:1
Enter question file name:(question.txt)
question.txt
*******Options*******************************
    1) Write answer into file
    2) Show answer on screen
    3) Exit
Enter your option:2
Question: What is the location of quarry X

WSML Query:

?x[hasLocation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Location
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry
?x [hasName hasValue X]

Question: What is the adress of industrial site X ?

WSML Query:

?x[hasAdress hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Address
annot#annotate(?x, ?x1) and
?x1 memberOf domain#IndustrialSite
?x [hasName hasValue X]

Question: What is the population of regional X ?

WSML Query:

?x[hasPopulation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Population
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Regional
?x [hasName hasValue X]

Question: Who mandates construction company Y ?

WSML Query:

?x[mandates hasValue ?y] and
annot#annotate(?y, ?y1) and
```

```
?y1 memberOf domain#ConstructionCompany
annot#annotate(?x, ?x1) and
?x1 memberOf domain#ConstructionSiteOwner
?y [hasName hasValue Y]
```

Question: What is the industrial activity of quarry X ?

WSML Query:

```
?x[hasIndustrialActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Exploitation
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry
?x [hasName hasValue X]
```

Question: What is the population of country Y ?

WSML Query:

```
?x[hasPopulation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Population
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Country
?x [hasName hasValue Y]
```

Question: What is the production rate of exploitation Z ?

WSML Query:

```
?x[hasProductionRate hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#ProductionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Exploitation
?x [hasName hasValue Z]
```

Question: What is the price of quarry product Y ?

WSML Query:

```
?x[hasPrice hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#AggregatePrice
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryProduct
?x [hasName hasValue Y]
```

Question: What is the functional property of quarry product Y ?

WSML Query:

```
?x[hasFunctionalProperty hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#FunctionalProperty
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryProduct
?x [hasName hasValue Y]
```

Question: What is determined by rock type X ?

WSML Query:

```
?x[determinedBy hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#RockType
annot#annotate(?x, ?x1) and
?x1 memberOf domain#MineralResource
?y [hasName hasValue X]

Question: What determines allowed production rate ?

WSML Query:

?x[determines hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#AllowedProductionRate
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Legislation
?y [hasName hasValue rate]

Question: What activity involves activity sawing ?

WSML Query:

?x[involvesActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Sawing
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Extraction
?y [hasName hasValue involves]

Question: Who administrates quarry x ?

WSML Query:

?x[administrates hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Quarry
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryAdministration
?y [hasName hasValue x]


*******Options*******************************
   1) Read questions from file
   2) Read questions from keyboard
   3) Exit
Enter your option:2
Enter your question:
What is determined by rock type X ?
*******Options*******************************
   1) Write answer into file
   2) Show answer on screen
   3) Exit
Enter your option:2
Question: What is determined by rock type X ?

WSML Query:

?x[determinedBy hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#RockType
annot#annotate(?x, ?x1) and
?x1 memberOf domain#MineralResource
```

```
?y [hasName hasValue X]

*******Options*******************************
    1) Read questions from file
    2) Read questions from keyboard
    3) Exit
Enter your option:2
Enter your question:
Who administrates quarry x ?
*******Options*******************************
    1) Write answer into file
    2) Show answer on screen
    3) Exit
Enter your option:2
Question: Who administrates quarry x ?

WSML Query:

?x[administrates hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Quarry
annot#annotate(?x, ?x1) and
?x1 memberOf domain#QuarryAdministration
?y [hasName hasValue x]


*******Options*******************************
    1) Read questions from file
    2) Read questions from keyboard
    3) Exit
Enter your option:2
Enter your question:
What is the population of country Y ?
*******Options*******************************
    1) Write answer into file
    2) Show answer on screen
    3) Exit
Enter your option:2
Question: What is the population of country Y ?

WSML Query:

?x[hasPopulation hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Population
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Country
?x [hasName hasValue Y]


*******Options*******************************
    1) Read questions from file
    2) Read questions from keyboard
    3) Exit
Enter your option:2
Enter your question:
What is the industrial activity of quarry X ?
*******Options*******************************
    1) Write answer into file
    2) Show answer on screen
    3) Exit
Enter your option:2
Question: What is the industrial activity of quarry X ?
```

```
WSML Query:

?x[hasIndustrialActivity hasValue ?y] and
annot#annotate(?y, ?y1) and
?y1 memberOf domain#Exploitation
annot#annotate(?x, ?x1) and
?x1 memberOf domain#Quarry
?x [hasName hasValue X]


*******Options*******************************
    1) Read questions from file
    2) Read questions from keyboard
    3) Exit
Enter your option:
```

Appendix D – ONALDIS Code in Java

This appendix contains the program code of ONALDIS.

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;

import easyIO.In;
import easyIO.Out;


public class ReadingOntology {

	HashMap nodeHashMap = new HashMap();
	static HashMap patterns = new HashMap();
	public static void main(String[] args) {
		String[] filenames=args;

		try {
			patterns = new ReadingOntology().doProcess(filenames);
		}
		catch (Exception e) {
			e.printStackTrace();
		}
	}
	public HashMap doProcess(String[] filename) throws Exception{
		for(int i = 0; i < filename.length; i++){
			readOntologyFile(filename[i]);
		}
		handleSuperClass();
		generatPatterns();
		writenPatternsToFile("ontologyPattern.txt");

		Iterator it = patterns.values().iterator();
		int i =0;
		while(it.hasNext()){
			i++;
			String wsml= (String)it.next();
			System.out.println(i+") \n"+wsml+"\n");
		}
		it = patterns.keySet().iterator();
		  i =0;
		while(it.hasNext()){
			i++;
			String pattern= (String)it.next();
			System.out.println(i+") "+pattern);
		}
		return patterns;
	}
	private void writenPatternsToFile(String filename) {
		Out resultFile=null;
		resultFile = new Out(filename);
		Iterator it = patterns.keySet().iterator();

		while(it.hasNext()){
			resultFile.outln(it.next());
			resultFile.outln("#");
		}
		it = patterns.values().iterator();
		while(it.hasNext()){
			resultFile.outln(it.next());
			resultFile.outln("#");
		}
		resultFile.close();

	}
	public void readOntologyFile(String filename) throws Exception{
```

```
Node fromN = null;
In file= new In(filename);
while(!file.endOfFile()) {
        String line = file.inLine();
        String readConcept = "";
        String readRelation = "";
        String relatedConcept = "";
        Node toN = null;
        if(line != null && line.length()>1){
                while(line.length()>1 && (line.substring(0, 1).equals("\t") || line.substring(0, 1).equals(" ")) ||
        line.substring(0, 1).equals("(")){
                        line = line.substring(1);
                }
                String [] wordsInLine = new String [line.split(" ").length];
                wordsInLine = line.split(" ");
                if(wordsInLine.length > 1 && wordsInLine[0].equals("concept")){
                        readConcept = wordsInLine[1];
                        fromN = (Node)nodeHashMap.get(readConcept);
                        if(fromN == null){
                                fromN = new Node(readConcept);
                                nodeHashMap.put(readConcept, fromN);
                        }

                        // subConcept
                        if(wordsInLine.length > 3 && wordsInLine[2].equals("subConceptOf")){
                                if(wordsInLine[3].contains("#")){
                                        wordsInLine[3] = wordsInLine[3].split("#")[1];
                                }
                                Node superN = (Node)nodeHashMap.get(wordsInLine[3]);
                                if(superN == null){
                                        superN = new Node(wordsInLine[3]);
                                        nodeHashMap.put(wordsInLine[3], superN);
                                }
                                fromN.addSuperNode(superN);
                        }
                }
                else{
                        if(line != null){
                                wordsInLine = line.split(" ");
                                if(wordsInLine.length > 1 && wordsInLine[1].equals("impliesType")&&
                                 !wordsInLine[0].substring(0, 2).equals("//")){
                                        readRelation = wordsInLine[0];
                                        if(wordsInLine[2].charAt(0) == '{'){
                                                String relatedConceptGroup =
                                                line.substring(line.indexOf("{"), line.indexOf("}")+1);
                                                        if(relatedConceptGroup.indexOf("#") != -1){
                                                                String relatedConceptGroup2 =
                                                line.substring(line.indexOf("{")+1, line.indexOf("}"));
                                                                String [] wordsinRelatedConcept =
                                                new String[relatedConceptGroup2.split(",").length];
                                                        wordsinRelatedConcept =
                                                        relatedConceptGroup2.split(",");
                                                        relatedConceptGroup+="{";
                                                for(int i = 0; i < wordsinRelatedConcept.length; i++){
                                                        wordsinRelatedConcept[i].replace(" ", "");
                                                wordsinRelatedConcept[i] = wordsinRelatedConcept[i].split("#")[1];
                                                 relatedConceptGroup += wordsinRelatedConcept[i];
                                                        if(i < wordsinRelatedConcept.length-1){
                                                                relatedConceptGroup += ",";
                                                                }
                                                        }
                                                }
                                                toN = (Node)nodeHashMap.get(relatedConceptGroup);
                                                if(toN == null){
                                                        toN = new Node(relatedConceptGroup);
                                                        nodeHashMap.put(relatedConceptGroup, toN);
                                                }
                                                fromN.addRelationToNodes(readRelation, toN);
                                                int numberOfConcepts = relatedConceptGroup.substring(1,
                                                relatedConceptGroup.length()).split(", ").length;
                                                String []concepts = new String[numberOfConcepts];
                                                concepts = relatedConceptGroup.substring(1,
                                                relatedConceptGroup.length()-1).split(", ");
                                                for(int j = 0; j < numberOfConcepts; j++){
                                                        relatedConcept = concepts[j];
                                                        if(relatedConcept.contains("#")){
                                                                relatedConcept =
                                                                relatedConcept.split("#")[1];

                                                        if((Node)nodeHashMap.get(relatedConcept)== null){

                                                nodeHashMap.put(relatedConcept, new Node(relatedConcept));
                                                                }
```

```
                                                ((Node)nodeHashMap.get(relatedConcept)).addSuperNode(toN);
                                                }
                                        }
                                }
                                else{
                                        relatedConcept = wordsInLine[2];
                                        if(relatedConcept.contains("#")){
                                                relatedConcept = relatedConcept.split("#")[1];
                                        }
                                        toN = (Node)nodeHashMap.get(relatedConcept);

                                        if(toN == null){
                                                toN = new Node(relatedConcept);
                                                nodeHashMap.put(relatedConcept, toN);
                                        }
                                        fromN.addRelationToNodes(readRelation, toN);
                                }
                        }
                }
        }
        file.close();
    }
    private void handleSuperClass() {
        Node n = null;
        ArrayList superNodes = null;
        Iterator it = nodeHashMap.values().iterator();
        while(it.hasNext()){
                n= (Node)it.next();
                if(n!=null){
                        superNodes = n.getSuperNodes();
                        while(!superNodes.isEmpty()){
                                Node superNode = (Node)superNodes.remove(0);
                                Iterator it2 = superNode.getRelationToNodes().keySet().iterator();
                                while(it2.hasNext()){
                                        String relation = (String)it2.next();
                                        n.addRelationToNodes(relation,
(Node)superNode.getRelationToNodes().get(relation));
                                }
                        }
                }
        }
    }
    private void generatPatterns() {
        Node n = null;

        Iterator it = nodeHashMap.values().iterator();
        while(it.hasNext()){
                n= (Node)it.next();
                if(n!=null){
                        HashMap relations = n.getRelationToNodes();
                        Iterator it2 = relations.keySet().iterator();
                        while(it2.hasNext()){
                                String relation = (String)it2.next();
                                String pattern = "";
                                String relatedNoteName = ((Node)n.getRelationToNodes().get(relation)).getNodeName();
                                String attr= "";
                                attr+= relation;
                                //pattern1, a hasXX b
                                if(relation.substring(0, 3).equals("has")){
                                        String relationWords = splitWords(relation.substring(3));

                                        pattern = "What .*" + relationWords + " of " + splitWords(n.getNodeName())+" .*";
                                        if(pattern.indexOf("//") != -1){
                                                pattern = pattern.split("//")[0];
                                                pattern += " .*";
                                        }

                                        String wsml = "?x["+attr+" hasValue ?y] and\nannot#annotate(?y, ?y1) and\n?y1
                                        memberOf domain#"
                                        +relatedNoteName+"\nannot#annotate(?x, ?x1) and \n?x1 memberOf
                                        domain#"+n.getNodeName();
                                        patterns.put(pattern, wsml);
                                }
                                //patter2
                                else{
                                        String relatedString =
                                        ((Node)n.getRelationToNodes().get(relation)).getNodeName();
                                        if(relatedString.indexOf("{") != -1){
                                                relatedString =
```

```
                                                relatedString.substring(relatedString.indexOf("{")+1,relatedString.indexOf("}"));
                                                        String [] wordinString = new String [relatedString.split(",").length];
                                                        wordinString = relatedString.split(",");
                                                        for(int i = 0; i < wordinString.length; i++){
                                                                wordinString[i] = wordinString[i].replace(" ", "");
                                                        }
                                                        for (int j = 0; j < wordinString.length; j++){
                                                                pattern = ".*" + splitWords(relation) + " " +
                                                                splitWords(wordinString[j])+".*";
                                                                String wsml = "?x["+attr+" hasValue ?y]
                                                                and\nannot#annotate(?y, ?y1) and\n?y1 memberOf
                                                                domain#"
                                                                        + wordinString[j] + "\nannot#annotate(?x, ?x1)
                                                                and \n?x1 memberOf domain#"+ n.getNodeName();
                                                                patterns.put(pattern, wsml);
                                                        }
                                                }
                                                else{
                                                        pattern = ".*" + splitWords(relation) + " " +
                                                splitWords(((Node)(n.getRelationToNodes().get(relation))).getNodeName())+" .*";
                                                if(pattern.indexOf("//") != -1){
                                                        pattern = pattern.split("//")[0];
                                                        pattern += ".*";
                                                }
                                                String wsml = "?x["+attr+" hasValue ?y] and\nannot#annotate(?y, ?y1) and\n?y1
                                                memberOf domain#"
                                                + relatedNoteName + "\nannot#annotate(?x, ?x1) and \n?x1 memberOf
                                                domain#"+ n.getNodeName();

                                                patterns.put(pattern, wsml);
                                                }
                                        }
                                }
                        }
                }
                private String splitWords(String relation) {
                        String relationWords = "";
                        for(int i = 0; i < relation.length(); i++){

                                if(relation.charAt(i) >= 'A' && relation.charAt(i) <= 'Z'){
                                        if(!relationWords.equals("")){
                                                relationWords += " ";
                                        }
                                        relationWords += relation.substring(i, i+1).toLowerCase() ;
                                }
                                else
                                        relationWords += relation.substring(i, i+1);
                        }
                        return relationWords;
                }
}


import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import easyIO.In;
import easyIO.Out;


public class MappingApplication {
        HashMap patterns = new HashMap();
        public static void main(String[] args) {

                MappingApplication mappingApplication = new MappingApplication();
                mappingApplication.menu(args[0]);
        }
        private void menu(String filename) {

                boolean notFinish = true;
                String questionFileName = "";
                String []questionArray = null;

                try {
                        readKeywordSetsAndWSMLQueryGenFromOnto(filename);
                } catch (Exception e1) {
                        e1.printStackTrace();
                }
                while(notFinish){
```

```
                System.out.print("*******Options*****************************\n" +
                                                        "       1) Read questions from file\n" +
                                                        "       2) Read questions from keyboard\n" +
                                                        "       3) Exit\n" +
                                                        "Enter your option:");
                In in = new In();
                int option = in.inInt();
                if(option == 1){
                        System.out.println("Enter question file name:(question.txt)");
                        questionFileName = in.inWord();
                        try {
                                questionArray = readQuestionFromFile(questionFileName, questionArray);
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
                else if(option == 2){
                        System.out.println("Enter your question:");
                        String question = in.inLine();
                        questionArray = new String[1];
                        questionArray[0] = question;
                }
                else if(option == 3){
                        return;
                }
                else{
                        System.out.print("Please enter the right number. Try again>>\n\n");
                }
                System.out.print("*******Options*****************************\n" +
                                        "       1) Write answer into file\n" +
                                        "       2) Show answer on screen\n" +
                                        "       3) Exit\n" +
                                        "Enter your option:");
                Out resultFile=null;
                int answerOption = in.inInt();
                if(answerOption == 1){
                        System.out.println("Enter result file name: (result.txt)");
                        String resultFilename = in.inWord();
                        resultFile = new Out(resultFilename);
                        mapping(questionArray, resultFile, patterns);
                        resultFile.close();
                        System.out.println("The wsml queries have been written into the result file.");
                }
                else if(answerOption == 2){

                        mapping(questionArray, resultFile, patterns);
                        System.out.println();
                }
        }

}
private void readKeywordSetsAndWSMLQueryGenFromOnto(String filename) throws Exception {
        In file= new In(filename);
        ArrayList lines = new ArrayList();
        String entry = "";
        while(!file.endOfFile()) {
                String line = file.inLine();
                if(!line.equals("#")){
                        entry+=line;
                        entry+="\n";
                }
                else{
                        entry= entry.substring(0, entry.length()-1);
                        lines.add(entry);
                        entry = "";
                }
        }
        for(int i = 0; i < lines.size()/2; i++){
                patterns.put(lines.get(i), lines.get(lines.size()/2+i));
        }
}
String[] readQuestionFromFile(String fileName, String []questionArray) {
        In file = null;
        try{
        file = new In(fileName);
        }
        catch(Exception e){
                do{
                In in = new In();
                System.out.println("Enter file name:");
                fileName = in.inWord();

                try {
```

```
                        file= new In(fileName);
                } catch (Exception e1) {

                }
                }while(file==null);
        }
        int lineNumber = 0;
        while(!file.endOfFile()) {
                file.inLine();
                lineNumber ++;
        }
        file.close();
        In file2=null;
        try{
                file2 = new In(fileName);
        }
        catch (Exception e){

        }
        questionArray = new String[lineNumber];
        int currentLine = 0;
        while(!file2.endOfFile()) {
                String line = file2.inLine();
                questionArray[currentLine++] = line;
        }

        file2.close();
        return questionArray;
}

private void mapping(String[] questionArray, Out resultFile, HashMap patterns) {

        for(int i = 0; i < questionArray.length; i++){
                String question = questionArray[i];

                boolean foundMatch = false;
                String wsml;
                Iterator it = patterns.keySet().iterator();
                while(it.hasNext()){

                        String keywordList = (String) it.next();
                        Pattern p = Pattern.compile(keywordList, Pattern.CASE_INSENSITIVE);
                        Matcher m = p.matcher(question);
                        if(m.matches()==true){
                                foundMatch = true;

                                wsml = (String) patterns.get(keywordList);
                                String name = "";
                                String wordBeforeName = keywordList.split(" ")[keywordList.split(" ").length-2];
                                String [] wordsInQuestion = new String[question.split(" ").length];
                                wordsInQuestion = question.split(" ");
                                for(int k = 0; k < wordsInQuestion.length; k++){
                                        if(wordsInQuestion[k].equals(wordBeforeName)){
                                                name = wordsInQuestion[k+1];
                                                break;
                                        }
                                }
                                String x = "";
                                if(keywordList.indexOf("of")!= -1){
                                        x = "x";
                                }
                                else x = "y";
                                wsml += "\n?"+ x +" [hasName hasValue " + name + "]";

                                if(resultFile!=null){
                                        resultFile.outln("Question: "+question);
                                        resultFile.outln("\nWSML Query: \n");
                                        resultFile.outln(wsml);
                                        resultFile.outln();
                                }
                                else{
                                        System.out.println("Question: "+question);
                                        System.out.println("\nWSML Query: \n");
                                        System.out.println(wsml);

                                        System.out.println();
                                }
                                break;
                        }
                }
                if(!foundMatch){
                        if(resultFile!=null){
                                resultFile.outln("\tno match");
```

```
                                        }
                                        else
                                        System.out.println("Sorry, the program can not answer your question, the reason might be the
following:\n      1) The words in question are wrongly spelled.\n      2) The question is out of the specified domain.\n\nPlease try other questions.
");
                                    }
                                }
                            }
}

import java.util.ArrayList;
import java.util.HashMap;

public class Node{

    private String nodeName;
    private HashMap relationToNodes = new HashMap();
    private ArrayList superNodes = new ArrayList();


  public Node() {

    }
    public Node(String nodeName) {
        this.nodeName = nodeName;
    }
    public void addRelationToNodes(String relationName, Node toNode){
            relationToNodes.put(relationName, toNode);
    }
    public HashMap getRelationToNodes(){
            return relationToNodes;
    }
    public void addSuperNode(Node n){
            superNodes.add(n);
    }
    public ArrayList getSuperNodes(){
            return superNodes;
    }
    String   getNodeName(){
            return nodeName;
    }
}
```

References

1. Solheim Ida, Stølen Ketil. Technology Research Explained SINTEF. A313 SINTEF report, March 2006.
2. Berners-Lee, T., James Hendler, Ora Lassila The Semantic Web. Scientific American Magazine, May 17, 2001.
3. W3C, W3C Semantic Web Frequently Asked Questions. 2008-03-13.
4. W3C, W3C Semantic Web Activity, http://www.w3.org/2001/sw/
5. InfoMesh, The Semantic Web: An Introduction
6. W3C, Semantic Web, and Other Technologies to Watch. 2008-05-13.
7. Wikipedia, URI. http://en.wikipedia.org/wiki/Uniform_Resource_Identifier
8. W3C, RDF. http://www.w3.org/RDF/
9. W3C, OWL. http://www.w3.org/2004/OWL/
10. Tim Berners-Lee, James Hendler and Ora Lassila, The Semantic Web, Scientific American.com, May 17, 2001.
11. W3C, RDFS. http://www.w3.org/TR/rdf-schema/
12. Yuxiao Zhao, K.S., Potential Advantages of Semantic Web for Internet Commerce, Dept of Computer and Information Science, Linkøping University.
13. Newcomer, E., Understanding Web Services: XML, WSDL, SOAP, and UDDI. 2002, Addison-Wesley Professional.
14. K.Gottschalk, S.Graham, H.Kreger, J.Snell, Introduction to Web services architecture.
15. Cerami, E., Web Services Essentials, O'Reilly Media, Februry 2002.
16. Olaf Zimmermann, M.T., Stefan Peuser, Perspectives on Web Services - Applying SOAP, WSDL and UDDI to Real-World Projects. Springer, september 26, 2005.
17. W3C, Web Services Description Language (WSDL) 1.1. 2001. http://www.w3.org/TR/wsdl
18. Havey, M., Essential Business Process Modeling. O'ReILLY, August 2005.
19. OASIS, W3C Acknowledges Receipt of Web Service Choreography Interface (WSCI) Submission, W3C, June 2002.
20. Wikipedia, BEPL. http://en.wikipedia.org/wiki/Business_Process_Execution_Language
21. W3Schools, SOAP Tutorial. http://www.w3schools.com/soap/default.asp
22. Cardoso, J., Semantic Web Services and Web Process Composition. Springer, July 2004.
23. Parsia, B., Semantic Web Services Interest Group. http://www.w3.org/2002/ws/swsig/
24. W3C, OWL-S Member submission http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/
25. Semwebcentral, The OWL-S Editor http://www.semwebcentral.org/projects/owlseditor/
26. Semwebcentral, The OWL-S Matcher http://owlsm.projects.semwebcentral.org/
27. Thomas R.Grimes, Truth, Content, and the Hypothetico-Deductive Method, The university of Chicago Press, 1990.
28. W3C, WSMO Member submission http://www.w3.org/Submission/WSMO/
29. WSMO org, WSML http://www.wsmo.org/wsml/
30. Vanessa Lopez, E.M., Ontology-driven Question Answering in AquaLog. Knowledge Media Insitute & Centre for Research in Computing,The Open University, 2004.
31. DERI., WSML 2 Reasoner Framework, http://tools.sti-innsbruck.at/wsml2reasoner/
32. Sourceforge, WSMO4j. http://wsmo4j.sourceforge.net/
33. W3C, WSDL-S Member submission http://www.w3.org/Submission/WSDL-S/

34. Wikipedia, Question answering. http://en.wikipedia.org/wiki/Question_answering
35. MIT, Start Question Answering System. http://start.csail.mit.edu/
36. Costa, L., A modular question answering system for Portuguese, Linguateca at SINTEF ICT, 2006.
37. Andrew Hickl, P.W., John Lehmann, and Sanda Harabagiu, FERRET: Interactive Question-Answering for Real-World Environments. 2006.
38. Sven Hartrumpf, H.H., Rainer Osswald, The semantically based computer lexicon HaGenLex – Structure and technological environment, Association pour le traitement automatique des langues, 2003.
39. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. 2002.
40. Tablan, V., Maynard, D., Bontcheva, K., GATE - A Concise User Guide.
41. University, P., WordNet. http://wordnet.princeton.edu/
42. Wikipedia，Semantic Web Services, http://en.wikipedia.org/wiki/Semantic_Web_Services.
43. Swing-project.org，SWING, http://www.swing-project.org/
44. Jinghai Rao, Dimitar Dimitrov, Paul Hofmann and Norman Sadeh, A Mixed Initiative Semantic Web Framework for Process Composition, School of Computer Science, Carnegie Mellon University
45. Richardson M., Midwinter T., Product Catalogue Management with Semantic Web Services, Springer Netherlands, November 2006.